Probabilistic Conflict Detection for Commercial Aircraft near Airports

Leanne Jane Pienaar

Electronic Systems Laboratory, Department Electrical and Electronic Engineering,
Stellenbosch University, Stellenbosch, South Africa, 7600

Supervisor:

Prof. T. Jones

Thesis presented in partial fulfilment of the requirements for the degree of Master in Engineering at Stellenbosch University

March 2015

Declaration

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the owner of the copyright thereof (unless to the extent explicitly otherwise stated) and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

November 2014

Abstract

Increasing air traffic and urbanisation has led to a cluttered airspace, particularly near airports, where both complex terrain and multiple moving obstacles are frequent. Accurately and efficiently predicting violations in safe separation criteria for commercial aircraft, a process called conflict detection, is therefore crucial in assessing risk associated with threats of collision. Existing avoidance systems in operation such as TCAS, EGPWS and ATC exhibit shortcomings, leaving room for uncertainty and possible conflict scenarios. A single on-board system capable of minimising errors in prediction would inform conflict resolution decisions more accurately as well as support the notion of free flight, an objective of next-generation air traffic management systems.

This thesis investigates the viability of a modern algorithm, probability flow, as a method of probabilistic conflict detection for commercial aircraft in airport environments. Simulation results for realistic flight scenarios are presented in comparison with a ground-truth result obtained through Monte Carlo simulation. Observations are made regarding the suitability of probability flow for real-world application. It is found that probability flow is capable of calculating a tight upper bound to the probability of conflict quickly and accurately for most conflict scenarios. However, unreasonably large overestimates on the probability of conflict are obtained when flying parallel to an obstacle conflict region. This problem could lead to a high frequency of false alerts, particularly in aborted landing scenarios and at airports operating parallel runways. It is therefore advised that further research be conducted to resolve this problem before probability flow can be reliably implemented in an airport environment.

Opsomming

Toenemende lugverkeer en verstedeliking het gelei tot ‘n deurmekaar lugruim, veral naby lughawens, waar byde komplekse terrein en verskeie bewegende struikelblokke gereeld voorkom. Akkuraat en doeltreffende voorspelling van oortredings in veilige skeidingskriteria vir kommersiële vliegtuie, naamlik konflik opsporing, is dus van kardinale belang in die beoordeling van die risiko wat verband hou met dreigemente van ‘n botsing. Bestaande vermyding stelsels in werking soos TCAS, EGPWS en ATC toon tekortkominge, wat ruimte laat vir onsekerheid en moontlike konflik scenario’s. ‘n Enkele aanboordstelsel, wat in staat is om foute in voorspelling te verminder, sou konflikresolusie besluite meer akkuraat in kennis stel, asook om die idee van vrye vlug te ondersteun, ‘n doelwit van toekomstige lugverkeer beheerstelsels.

Hierdie tesis ondersoek die lewensvatbaarheid van ‘n moderne algoritme, waarskynlikheidsvloei, as ‘n metode van probabilistiese konflik opsporing vir kommersiële vliegtuie in die lughawens omgewing. Simulasie resultate vir realistiese vlug scenario’s word aangebied in vergelyking met ‘n grond-waarheid resultaat wat verkry word deur middel van Monte Carlo simulasie. Waarnemings word gemaak ten opsigte van die geskiktheid van waarskynlikheidsvloei vir die werklikheid. Dit is bevind dat waarskynlikheidsvloei in staat is om die berekening van ‘n stywe bogrens tot die waarskynlikheid van konflik vinnig en akkuraat te bepaal vir die meeste konflik scenario’s. Tog is daar ‘n onredelike groot oorskatting op die waarskynlikheid van konflik wat verkry word wanneer ‘n vliegtuig parallel met ‘n hindernis konflik streek vlieg. Hierdie probleem kan lei tot ‘n hoë frekwensie van valse waarskuwings, veral in mislukte landing scenario’s en by lughawens wat van parallel aanloopbane gebruik maak. Dit word dus aanbeveel dat verdere navorsing gedoen word om die probleem op te los voordat waarskynlikheidsvloei betroubaar in ’n lughawe omgewing geïmplementeer word.

Acknowledgements

I would like to thank the Airbus Group and the South African National Aerospace Centre for funding this research. I would also like to thank Sophie Lambeaux and Pierre Fabre along with their colleagues at Airbus Toulouse for their technical guidance and feedback. Much gratitude is also extended to the project supervisor, Thomas Jones, as well as Corné van Daalen and Japie Engelbrecht for their valuable insight.

Contents

List of Figures

List of Tables

Nomenclature

Abbreviations

ft	Feet
GB	Gigabytes
KB	Kilobytes
km	Kilometres
km/h	Kilometres per hour
m	Metres
m/s	Metres per second
MB	Megabytes
MHz	Megahertz
NM	Nautical miles
s	Seconds

Acronyms

2D	Two-dimensional
3D	Three-dimensional
ACAS	Airborne Collision Avoidance System
ADS-B	Automatic Dependent Surveillance - Broadcast
API	Application Programmers Interface
ATC	Air Traffic Control
ATCO	Air Traffic Control Officer
ATM	Air Traffic Management
ATMS	Air Traffic Management System

AUV	Autonomous Underwater Vehicle
CFIT	Controlled Flight Into Terrain
Comm.	Communications
CPA	Closest Point of Approach
FAA	Federal Aviation Administration
FL	Flight Level
FLARM	Flight Alarm
GCC	GNU Compiler Collection
GIS	Geographic Information System
GPS	Global Positioning System
GPWS	Ground Proximity Warning System
HTTP	Hyper-Text Transfer Protocol
ICAO	International Civil Aviation Organisation
IFR	Instrument Flight Rules
IMU	Inertial Measurement Unit
JSON	JavaScript Object Notation
MSL	Mean Sea Level
MTOM	Maximum Take-Off Mass
NED	North-East-Down axis system
NextGen	Next Generation Air Transportation System
NTSB	National Transportation and Safety Board
OCAS	Obstacle Collision Avoidance System
PAPI	Precision Approach Path Indicator
PCAS	Portable Collision Avoidance System
PDF	Probability Density Function
RA	Resolution Advisory

RAM	Random Access Memory
RVSM	Reduced Vertical Separation Minima
SESAR	Single European Sky ATM Research
TA	Traffic Advisory
TAWS	Terrain Awareness Warning System
TCAS	Traffic Collision Avoidance System
TCP	Trajectory Change Point
UAT	Universal Access Transceiver
UFIT	Uncontrolled Flight Into Terrain
URL	Uniform Resource Locator
US	United States (of America)
VRF	Visual Flight Rules
XML	Extensible Mark-up Language

Symbol Conventions

x	Scalar
$\mathbf{x}$	Vector
$\mathbf{x}(t)$	Time-varying vector
X	Set
$\mathbf{X}$	Matrix
$\mathbf{X}(t)$	Time-varying matrix
$\mathbf{X}(t, \omega)$	Vector of random processes
$\overline{x}$	Mean of x

List of Symbols

α	Angle of attack
$\alpha/2$	Critical value
β	Angle of sideslip

ΔM	Distance between 2 points on a sphere
λ	Coordinate of longitude
$\mathbf{n_f}$	Normal vector to a triangular mesh face
$\mathbf{n_p}$	Normal vector at vertex of mesh
$\mathbf{p_{new}}$	New point in terrain mesh after Minkowski sum
$\mathbf{p_{old}}$	Old point in original terrain mesh before Minkowski sum
$\mathbf{p}$	Position vector of a point in space
$\mathbf{r_h}$	A host aircraft position vector
$\mathbf{r_i}$	An intruder aircraft position vector
$\mathbf{v}$	A triangle vertex.
Φ, Θ, Ψ	Roll, pitch and yaw angles respectively
ψ_{track}	Heading of the flight route (ground track)
σ_{MC}	Standard deviation of Monte Carlo generated trajectories
$\sigma_{PC(MC)}$	Standard deviation of the Monte Carlo probability of conflict
θ_{track}	Flight path angle of flight route
φ	Coordinate of latitude
C_m	Centre of mass
E_m	Error margin
g_d	Difference in altitude between source waypoint and aircraft
h_d	Distance in NE-plane from source waypoint to aircraft
K_ψ	Cross-track controller gain
K_θ	Altitude controller gain
L, M, N	Roll, pitch and yaw moments respectively
L_{track}	Distance from source waypoint to destination waypoint
N_{GMC}	Number of Geometric Monte Carlo simulations or trajectories
N_{MC}	Number of Monte Carlo simulations or trajectories

N_F	Number of mesh faces
N_O	Number of detected threats near host aircraft
N_R	Number of Google Elevation API requests
N_S	The number of samples along a simulated path i.e. $\frac{T_f}{T_s}$
N_V	Number of mesh vertices
P, Q, R	Roll, pitch and yaw rates (angular velocity) respectively
P_C	True probability of conflict
P_{C_T}	Net probability of conflict over a specified future time interval
r_E	Radius of the Earth
R_h	Set of host aircraft position vectors
R_i	Set of intruder aircraft position vectors
s_I	Function parameter value at which an intersection occurs
T_f	Total simulation time
T_s	Simulation sampling period
U, V, W	Linear velocity vector coordinates in the X_P, Y_P and Z_P axes respectively
W_{T_D}	Turbulence in the Down inertial axis
W_{T_E}	Turbulence in the East inertial axis
W_{T_N}	Turbulence in the North inertial axis
W_{T_P}	Turbulence in the lateral principal axis
X, Y, Z	Force vector coordinates in the X_P, Y_P and Z_P axes respectively
x_d	Along-track distance travelled
X_P	Longitudinal (roll) axis in the principal axis system
X_P	Normal (yaw) axis in the principal axis system
y_d	Cross-track error
Y_P	Lateral (pitch) axis in the principal axis system
z_d	Altitude error
$z_{\alpha/2}$	Z-score of Gaussian distribution at critical value $\alpha/2$

Chapter 1

Introduction

In this chapter, the topic of research is introduced through a discussion of the problem, the purpose of the research and the project objectives. A summary follows of the overall thesis navigation and structure.

1.1 Problem Description

Commercial airspace is becoming increasingly cluttered and more complex for pilots to navigate safely. This is primarily due to a rising demand for public air transport and a more densely populated terrain. Aircraft manufacturers are producing larger aircraft and airlines are acquiring more aircraft to improve transport capacity.

The International Civil Aviation Organisation (ICAO) requires that aircraft maintain a safe separation distance between themselves and obstacles. A violation of this criteria is called a conflict. If a conflict occurs, a collision is considered to be imminent. It is becoming increasingly difficult to avoid conflict between aircraft, especially near airports where air traffic is most concentrated, terrain is most cluttered and flight paths are most limited.

As a possible solution, aviation authorities have proposed that free flight be implemented in next generation air traffic management (ATM) systems such as SESAR and NextGen (discussed in section 2.5 on page 21). Free flight is a navigational concept in which aircraft operate independently of any centralised traffic control. Aircraft are able to plan and re-route paths dynamically while maintaining safe separation criteria automatically. Research efforts in the field of automatic conflict detection and resolution are therefore important if free flight objectives are to be achieved.

1.2 Purpose of Work

Automatic conflict resolution is a very desirable feature for commercial aircraft because it has the potential to avoid collisions where pilots may fail to do so. However, this shifts some responsibility from pilots to aircraft manufacturers. Resolution decisions are based on the accurate detection of a conflict in the future. Manufacturers therefore need to have high confidence in the prediction result before automatic resolution can be implemented.

A large amount of work has been done on conflict detection for the en-route phase of commercial flight [10], but little has been done for the phases of flight in close proximity to an airport. Furthermore, collision avoidance systems currently used in practice are not fully equipped to handle the level of uncertainty present in the prediction of future aircraft trajectories. This thesis therefore focuses exclusively on probabilistic conflict detection for commercial aircraft near the airport environment. Although this approach considers piloted flight, it is important to note that the methods and concepts discussed in this thesis are also applicable to autonomous vehicles.

1.3 Objectives

The research objectives for this project are as follows:

1. Develop an understanding of current air traffic management systems and collision avoidance systems in practice;
2. Investigate existing methods of probabilistic conflict detection;
3. Model the relevant aspects of the airport environment;
4. Develop a collision avoidance framework incorporating conflict detection;
5. Implement the chosen probabilistic conflict detection method in various, realistic, simulated flight scenarios;
6. Evaluate the performance and suitability of the chosen method in the civil aviation context;
7. Make recommendations for improvement.

1.4 Expected Results and Significance

The primary contribution of this thesis lies in the evaluation of the probability flow algorithm for probabilistic conflict detection in a civil aviation context. The algorithm

should be tested for uncertain, cluttered environments, particularly near airports where multiple moving obstacles and dangerous terrain are frequent.

Van Daalen shows that the algorithm is capable of performing real-time conflict prediction in relatively simple scenarios. This thesis attempts to discover the limitations of the algorithm in terms of execution time and accuracy for larger expanses of terrain and multiple dynamic intruders.

Simulation results are expected to reveal algorithm trade-off between accuracy and efficiency. However, it is predicted that a tractable improvement can be offered on existing collision avoidance systems. To the author's knowledge, this thesis presents the first application and testing of the probability flow method in the civil aviation context and more specifically, at an airport.

1.5 Thesis Overview

In chapter 1, the topic of research is introduced and a general problem description is provided. The purpose of this project, scope and expected significance are also discussed before the chapter is concluded with an overview of the thesis.

Chapter 2 presents a review of recent literature relevant to collision avoidance. The fundamentals of commercial flight are presented along with a discussion of regulated safe separation requirements for civilian aircraft. This chapter investigates sources of uncertainty and intent in the airport environment and describes the operation of various existing collision avoidance systems in practice. Furthermore, the goals of future air traffic management systems in Europe and the USA are examined to provide motive and context for how this research may contribute to the development of anticipated aviation technology. Previous methods of probabilistic conflict detection are then presented before the chapter is concluded with a number of collision case studies, assisting the choice of test simulation scenarios used later in chapter 5.

Modelling concepts important to the accurate portrayal of the environment in simulation are outlined in chapter 3. A collision avoidance framework is proposed preceding the discussion of specific terrain and aircraft modelling techniques. The reference systems and basic notation used in the aircraft model are also defined.

In chapter 4, the implementation of the Monte Carlo and probability flow methods is described. A brief overview of the testing software is given as well as a detailed account

of how disturbance is introduced into the simulation to mimic uncertainty and control systems are employed to imitate a pilot's influence on the aircraft. The implementation of the Monte Carlo method is presented along with a description of the practical considerations that were required when memory exhaustive data generation was performed. The mathematical definition of probability flow is subsequently provided accompanied by a discussion of numerical methods used to efficiently solve the complex integrals present in the definition. The chapter is rounded off with a proposed formulation for calculating the net probability of conflict for multiple conflict events.

The penultimate chapter, chapter 5, presents the results of 4 simulation scenarios key to evaluating the performance of Van Daalen's probability flow method. Observations are made regarding accuracy relative to the Monte Carlo generated probability of conflict as well as the uncertainty in prediction. Various characteristics of the algorithm are noted through the examination of the simulation results.

Chapter 6 concludes this thesis by providing an overview of the achieved research objectives, a summary of the simulation results and a review of the discovered limitations and performance factors relevant to the probability flow method. Lastly, recommendations are made for future improvement of the algorithm with regards to efficiency and minimisation of uncertainty.

Chapter 2

Literature Review

This chapter introduces important collision avoidance concepts that promote contextual understanding of the research objectives. An overview of conflict avoidance systems currently in practice is provided as well as a discussion on existing methods of probabilistic conflict detection.

2.1 Phases of Commercial Flight

During a single flight, a commercial aircraft typically moves through 7 different phases (see figure 2.1) that are collectively called a flight profile.

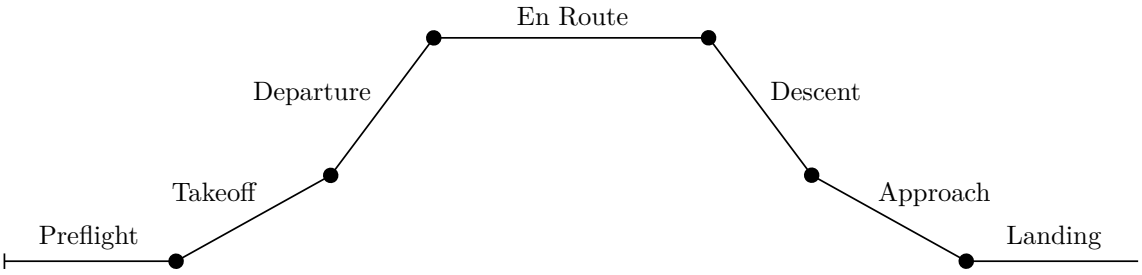


Figure 2.1: The 7 stages of a commercial flight profile adapted from NASA [1]

In this thesis, emphasis is placed on the first and last three phases of the flight profile, which are nearest the airport environment. In all cases, the aircraft under consideration is referred to as the host aircraft. Other aircraft and moving ground vehicles that may pose a threat to the host aircraft are referred to as intruders. Intruder aircraft and weather collectively form the dynamic environment while the static environment describes stationary obstacles such as terrain and protected airspace.

2.2 Safe Separation

In order to maintain safe flight, ICAO requires that minimum separation criteria be enforced between aircraft and obstacles. The protected airspace surrounding an aircraft is

referred to as the conflict region. A conflict therefore exists if any object violates the safe separation criteria. This is not to be confused with a collision, which occurs only if there is a physical impact between an aircraft and an obstacle.

Wake vortex turbulence is a contributing factor in the calculation of safe separation standards. It is made up of wing-tip vortices and jet-wash. Jet-wash is fast-moving gas expelled from an aircraft's engine, which causes short bursts of turbulence. Wing-tip vortices are more dangerous because they can remain in the air for up to 3 minutes after an aircraft has passed. When an aircraft's wing is experiencing lift, air from below the wing is forced around to the top of the wing where there is lower air pressure (see figure 2.2).

Figure 2.2: Wingtip Vortices

The effect of wake vortex turbulence is dependent on the weight and size of the aircraft. The Federal Aviation Administration (FAA) of the US and ICAO therefore categorise aircraft into different classes (see table 2.1).

Table 2.1: Aircraft Wake Vortex Classification

ICAO[11]	**MTOM***
Light	$\leq$ 7000 kg
Medium	7000 kg $<$ MTOM $<$ 136,000 kg
Heavy	$\geq$ 136,000 kg
FAA[12]	**MTOM**
Small	$\leq$ 19,000 kg
Large	19,000 kg $<$ MTOM $<$ 140,000 kg
Heavy	$\geq$ 140,000 kg
Super	Airbus A380 Only

* Maximum Take Off Mass

ICAO specifies different separation standards for each phase of commercial flight operating under Instrument Flight Rules (IFR). IFR require that the pilot navigate solely on the

basis of outputs from on-board instruments, especially in situations when weather may compromise visual awareness. Alternatively, pilots can fly using Visual Flight Rules (VFR) depending on the regulations of the country or airport airspace in which they are flying. VFR allow the pilot to navigate the aircraft by sight in conditions of good visibility. This relies on the pilot's ability to avoid obstacles by visual acquisition. Separation criteria for VFR are usually given by Air Traffic Control (ATC) or pilots are expected to maintain separation at their own discretion. IFR is used more often than VFR for commercial flights because routes frequently occupy clouded altitudes.

The following IFR separation rules are described by ICAO [11]:

En-route (Cruise)

- Vertical separation is 1000 ft (300 m) below an altitude of 29,000 ft (FL290) and 2000 ft (600 m) at or above FL290.
- Aircraft equipped with more advanced instruments may be allowed to follow Reduced Vertical Separation Minima (RVSM), which allows a 1000 ft (300 m) separation up to an altitude of 41,000 ft (FL410) and 2000 ft (600 m) between FL410 and 60,000 ft (FL600).
- 5000 ft (1500 m) vertical separation is enforced between all aircraft above FL600, regardless of RVSM.
- Aircraft following the same track require a longitudinal separation of 15 minutes at all times. For an aircraft travelling at a typical cruise speed of 475-500 knots, this equates to approximately 220-231 kilometres in longitudinal distance.
- Radar-controlled aircraft are required to be separated horizontally by at least 3 NM if within 40 NM of the radar antenna or by at least 5 NM if further than 40 NM from the antenna.

Departure and Descent

- An aircraft may immediately move to an altitude level previously occupied by another aircraft provided that the latter has reported departing it, severe turbulence is not present and minimum en-route vertical separation criteria are not violated.
- For aircraft departing or descending on the same track, a minimum separation of 5 minutes is required provided that the aircraft furthest from the airport is travelling 37 km/h faster than the aircraft closest to the airport. A minimum separation of only 3 minutes is required if the aircraft furthest from the airport is travelling 74 km/h faster than the aircraft closest to the airport.

- If two aircraft are departing on tracks that are at a 45 degree angle to one another at the divergence point, a 1 minute separation is recommended.
- Aircraft departing or descending on crossing paths are recommended to maintain a 15 minute separation from the intersection point, but it may be reduced to 10 minutes for aircraft equipped with modern navigation aids that permit frequent determination of speed and position.

Preflight and Take Off

- An aircraft in a lower class must wait 2 minutes before taking off following an aircraft of a higher class.
- A 3 minute waiting period is advised between aircraft taking off on parallel runways separated by 760 m or less. On parallel runways separated by more than 760 m, a 2 minute separation may be applied.
- Aircraft taking off from the same runway in opposite directions require a 2 minute separation.

Approach and Landing

- A minimum radar separation distance must be maintained between aircraft of different classes when landing (see table 2.2). Radar separation is used to regulate aircraft within 40 nautical miles (NM) of a radar antenna, usually at an airport.

Table 2.2: FAA Minimum Landing Radar Separation Criteria [6]

Preceding Aircraft	**Following Aircraft**	**Radar Separation (NM)**
Super	Super	2.5
	Heavy	6
	Large	7
	Small	8
Heavy	Heavy	4
	Large	5
	Small	6
Large	Large	2.5
	Small	4
Small	Small	2.5

There are also various regulations that govern specific regions of airspace. It is divided into classes A-G where A-E denotes controlled airspace and F-G is uncontrolled airspace. Controlled airspace is a predefined region in which air traffic control services are available. The degree of control is dependent on the specific class of airspace (see table 2.3). Uncontrolled airspace refers to regions, which may be too remote or high in altitude for air traffic control services to operate. Aircraft are independently responsible for their own air traffic navigation in these regions.

Table 2.3: Controlled Airspace Classes [7]

	Entry Requirement	**Navigation**	**Region**
A	ATC clearance, IFR equipped	IFR only	18,000 ft MSL* - FL600
B	ATC clearance, Two-way radio, Mode C/S transponder	IFR / VFR	10,000 ft MSL around a nation's busy airports
C	Two-way radio, Mode C/S transponder prior to entry	IFR / VFR	5 NM radius up to 1200 ft MSL and 10 NM up to 4000 ft MSL around airports with a radar control tower
D	Two-way radio prior to entry	IFR / VFR	2500 ft MSL around airports with a radar control tower
E	None for VFR	IFR / VFR	14,500 ft MSL up to 18,000 ft MSL

* Mean Sea Level

2.3 Intent and Uncertainty

It is important to know where an aircraft is expected to be in the future if a conflict is to be predicted. State variables contain information about the aircraft, such as position, heading and speed. The prediction of future aircraft states is dependent on the current states of the aircraft and the pilot's ability to follow a pre-determined flight route. A planned flight route forms part of the aircraft's intent, where intent refers to the planned states of the aircraft for a specific time interval in the future with particular interest in its position [13].

Regardless of whether or not the aircraft follows the planned flight route, a path that the aircraft is expected to follow in the future is known as a predicted trajectory. Knowledge about intended changes in future states improves the accuracy of a predicted trajectory. It is impossible to know what the exact intentions of the pilot are. It is therefore assumed

that the pilot follows the intended flight route within an envelope of uncertainty. It is possible that in the future, the pilot's intentions may be modelled to obtain a more accurate representation of the predicted trajectory, but this will require separate research and study in the area of predicting human behaviour. An aircraft under autopilot control behaves much like an ideal pilot, following the intended flight route, being only influenced by environmental factors such as wind, pressure, temperature and turbulence for example. In this thesis, the aircraft is therefore flown in simulation under autopilot control in conjunction with a turbulence model to imitate realistic conditions of uncertainty.

Uncertainty is not only present in the predicted trajectory of an aircraft, but also in the estimation of the current aircraft states as well as in maps of the terrain. State estimation uncertainty is primarily due to inaccuracies in on-board sensors or modelling errors in the dynamics of the aircraft. Errors in terrain maps exist because not every point of elevation on the Earth has been measured. Discrete terrain data is interpolated and further simplified depending on the format in which it is being stored. Moreover, uncertainty is also present in a system's knowledge of other aircraft states.

Reducing system uncertainty enables the avoidance system to be more confident in its prediction of conflict. Uncertainty can never be eliminated entirely and so, it should rather be incorporated into a conflict detection system than ignored. Uncertainty must therefore be modelled mathematically.

Paielli and Erzberger [14] chose to model uncertainties as Gaussian random variables for the along-track and cross-track errors. They managed to successfully obtain a closed-form analytical solution for a two-aircraft conflict scenario. Yang and Kuchar [2] further motivated this choice because it simplifies the problem considerably, but also argued that as the number and complexity of uncertainties increases in a trajectory model, it becomes difficult to obtain an accurate solution.

2.3.1 Trajectory Propagation

According to Yang and Kuchar [2], an aircraft's predicted trajectory can be obtained through three different propagation methods: nominal, worst-case and probabilistic. The nominal approach assumes that the aircraft trajectory follows a singular path. This is usually implemented by assuming that the aircraft flies in a straight line in the direction of the current velocity vector. This method is very simple, but does not incorporate any uncertainty. Kuchar and Yang also suggested that the nominal approach is not suitable because accurate predictions can only be calculated a few seconds into the future [10].

This is not appropriate for scenarios where aircraft are in the en-route phase of flight travelling at high speeds over a long distance. However, this could be sufficient for an airport scenario where conflict will be predicted only a few seconds in the future, but it is possible that missed detections will still be prevalent.

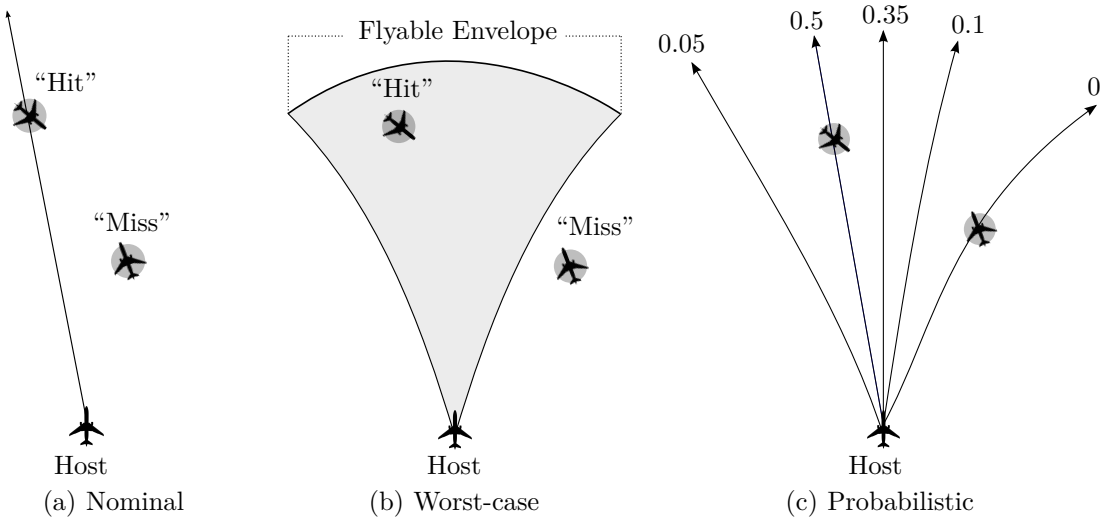


Figure 2.3: Methods of Trajectory Propagation [2]

Alternatively, an extension of nominal trajectory propagation called waypoint navigation can be used to predict the aircraft trajectory. Waypoints represent an ordered sequence of GPS coordinates along a proposed flight route. With waypoint navigation, it could be assumed that the aircraft is flying along straight-line segments between waypoints. It is then easy to predict the trajectory of the aircraft if it is known that it will pass through certain GPS coordinates. The disadvantage of this method is that the aircraft might not necessarily be flying perfectly in a straight line between waypoints due to wind and sensor inaccuracies, and pilots may deviate from their routes in the event of bad weather, emergency or malfunction.

To improve the waypoint navigation method, Yang et al. [2] developed a trajectory prediction model that specifies an expected time of arrival (ETA) at each waypoint. Additionally, the number of waypoints can be reduced to include only trajectory change points (TCPs), locations at which significant changes occur in the aircraft's states. This simplification was also used by Poretta et al. [15]. These TCPs mostly lie on points where heading changes are necessary or at the boundary between phases of flight (see figure 2.1 on page 5) where altitude changes occur. The TCPs are then ordered in terms of their associated ETA.

The worst-case method considers all possible paths that an aircraft is able to follow in a future time-interval. The possible paths are only limited by the physical capabilities of the aircraft. The nominal and worst-case propagation methods both return a binary result of true or false when checking for a conflict. This is a disadvantage for the worst-case method because highly improbable paths may cause a conflict to be detected. The worst-case method therefore results in many false alerts.

The probabilistic method models aircraft state information using random variables. Possible trajectories are weighted by their likelihood of occurrence. The aircraft's position, heading and velocity can for instance be modelled by their probability distributions. The probabilistic approach is taken in this thesis because it allows uncertainty to be incorporated into the trajectory propagation model and it offers the best trade-off between missed detections and false alerts.

2.3.2 Sensors

The first step in trajectory prediction is to determine the current states of the aircraft, such as speed, heading and position. This information can be derived from sensors and instruments on-board the aircraft.

The primary hardware responsible for determining aircraft state information is the Inertial Measurement Unit (IMU). The IMU consists of accelerometers and rate gyroscopes. There are usually three accelerometers in an IMU, which can measure acceleration of the aircraft in three different axes. There are also three rate gyroscopes, which measure the three-dimensional (3D) angular rates of the aircraft.

Unfortunately, the IMU can often accumulate errors because the navigation system continuously sums detected changes at each time instant with its previously measured values. As a solution, other sensors external to the IMU are used to check for errors and correct the measurements when necessary. In more sophisticated sensor installations, magnetometers are utilised to detect 3D magnetic field strength. This magnetic field strength is used to determine the heading of the aircraft.

A Global Positioning System (GPS) receiver is used on most modern civilian aircraft for its ability to accurately determine an aircraft's position in time as well as nearby weather information. GPS consists of multiple satellites orbiting the Earth and an aircraft must have an unobstructed line of sight to at least 4 satellites to receive GPS information. Radar is used to calculate the angle and range between an aircraft and its surroundings.

It achieves this by bouncing radio waves off objects in the environment and measuring the time it takes to receive the signal again. In this manner, the bearing and position of the aircraft in the inertial reference frame can be obtained.

A pitot tube is an instrument that can measure aircraft velocity (airspeed) and resides on the fuselage or wing of the aircraft. These three devices are used together with the IMU to determine the current states of the aircraft at every time instant during flight. Unfortunately, one can never be entirely certain about the integrity of state information, which is why it can be referred to as a state estimate.

2.3.3 Static Environment

In order to predict conflict with the terrain, precise information on the location and elevation of mountains, buildings, protected airspace and other stationary hazards is required. Currently, the best solution is to use a Geographical Information System (GIS), which includes a database containing geographical maps for regions worldwide. By 2012, ICAO had accumulated GIS maps for over 4 300 cities, 43 875 routes and 31 096 200 movements [16]. The system is still consistently expanding today.

In this thesis, the assumption is made that the host aircraft will always have access to a GIS map for the geographical regions encapsulating the intended flight path. Furthermore, we assume that the aircraft is equipped with radar, which assists in the identification of the aircraft's surroundings. The host aircraft is therefore considered to have full knowledge of the static environment.

2.4 Collision Avoidance Systems in Practice

Various collision avoidance systems are currently in operation on the ground and in the air on-board aircraft. An Airborne Collision Avoidance System (ACAS) operates independently of ground-based equipment and other aircraft. Its only concern is for the safety of the aircraft upon which it operates. If a threat is evident, the system provides the pilot with visual and aural warnings. If a collision is considered imminent, the system also advises the pilot on a manoeuvre to avoid the collision.

ACAS relies on information from on-board sensors and instruments to predict future collisions. Examples of ACAS are the Traffic Collision Avoidance System (TCAS), Ground Proximity Warning System (GPWS), Portable Collision Avoidance System (PCAS) and

Flight Alarm (FLARM) devices.

Ground-based systems such as Air Traffic Control (ATC) and the Obstacle Collision Avoidance System (OCAS) mostly rely on radar and Automatic Dependent Surveillance - Broadcast (ADS-B) interrogations to determine the location and trajectory of aircraft relative to obstacles. Advisories about possible threats of collision are communicated to pilots from the ground via radio or ADS-B. ACAS warnings are prioritised above those received from ground-based systems.

2.4.1 Traffic Collision Avoidance System

A mid-air collision between two airliners that occurred over the Grand Canyon in 1956 first revealed the need to civil aviation authorities for an effective collision avoidance system. Early developments identified that the solution should be cooperative and capable of issuing complementary and non-conflicting manoeuvre commands to all pilots involved in a close encounter. However, this requires a reliable communication link between aircraft. [8]

Design attempts by various airlines and manufacturers throughout the 1970s led to the first use of transponders as a means for aircraft to determine an intruder's range and altitude. A Mode A transponder is capable of responding to interrogations with a 4-digit Octal identifying ("squawk") code, but a Mode C transponder can additionally transmit aircraft pressure altitude in 100 ft increments measured from an on-board barometer. Only 4,096 identity codes exist in Mode A and C making it difficult to keep identity assignments unique in crowded airspace. The squawk code of the aircraft is manually set by the pilot in the cockpit, but only at the request of an air traffic control officer (ATCO).

At first, the accuracy and reliability of transponders was poor due to a cluttered radio frequency band and a limited number of squawk codes, especially for high traffic areas. Many false alarms resulted in the testing of initial systems. Another mid-air collision that occurred in 1978 above San Diego spurred the FAA to concentrate efforts on developing the first widely used avoidance system for commercial aircraft, TCAS. [8]

TCAS I is the first and most basic form of ACAS. It monitors surrounding air traffic and issues a "traffic, traffic" warning to the pilot if an intruder is nearby. TCAS I does not offer a solution to resolve the threat and only assists the pilot in the visual acquisition of other aircraft. Once a threat is no longer imminent, TCAS I issues a "clear of conflict" alert. TCAS warnings and alerts are called traffic advisories (TAs). TCAS II, TCAS I's

Table 2.4: TCAS II Alarm Thresholds [8]

Host Aircraft Altitude (feet)	**Time to Collision (seconds)**	
	TA	**RA**
< 1000	20	N/A
1000 - 2350	25	15
2350 - 5000	30	20
5000 - 10,000	40	25
10,000 - 20,000	45	30
> 20,000	48	35

successor, offers the added functionality of issuing vertical resolution advisories (RAs) to the pilot. An RA is a recommended exit manoeuvre to either maintain or increase vertical separation between aircraft. Both aircraft involved in a conflict are given complementary RAs to avoid aggravation of the situation. [8]

TCAS II was designed to cope with a traffic density of 24 aircraft within a 5 nautical mile radius, which is the maximum envisioned by the FAA for the next 20 years. Each conflict situation is considered one at a time and is prioritised based on time to the closest point of approach (CPA). The alarm time thresholds for TCAS II vary by altitude and are listed in table 2.4. An example of the TCAS separation regions is provided in figure 2.4 on the following page for the 5000 to 10,000 ft case.

2.4.1.1 Mode S transponders

By 1981 when TCAS I was starting to be implemented, Mode S (Select) transponders had been created. Mode S transponders are more advanced than their predecessors in that they use discrete address communication techniques, which permit more dependable coordinated escape manoeuvres between aircraft. [8]

Every aircraft is assigned a unique 24-bit address by ICAO, which is used to identify a particular aircraft when interrogated. The address consists of a header block, which identifies the country of origin, and the number of remaining bits determines how many allocations can be made in that country. For example, the UK has a 6-bit header block, which leaves 18 address bits corresponding to 2^{18} (262,144) registered aircraft. Austria has a 9-bit header block, meaning they can only have 32,768 codes to allocate. The number of allocations per country is regulated by ICAO based on the size of the country and amount

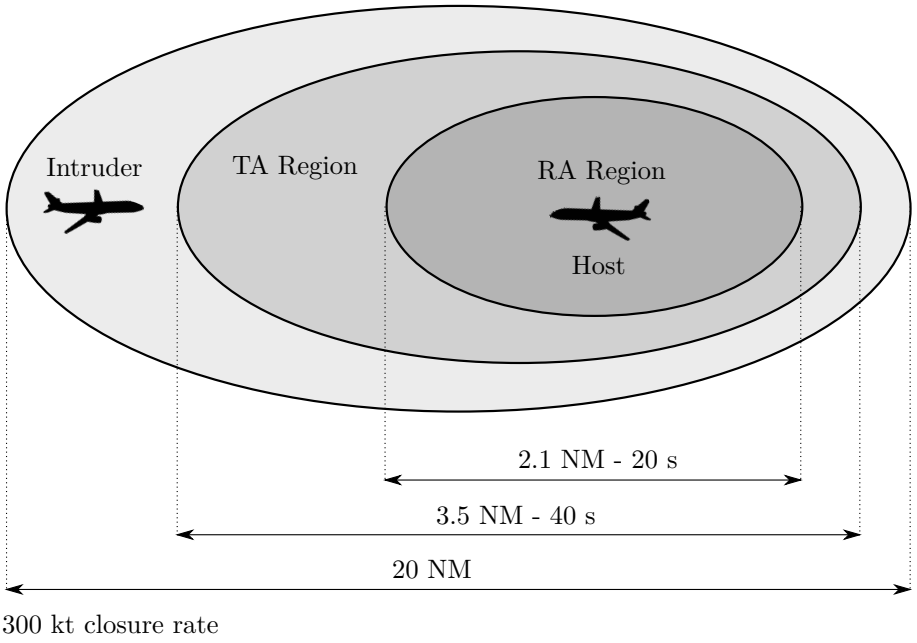


Figure 2.4: Example of TCAS separation regions between 5000 and 10,000 feet [3]

of air traffic in the region. [17]

Each time a transponder is interrogated, an aircraft replies with a message that consists of an 8 µs preamble ("handshake") followed by a data block. A short data block is made up of a 5-bit format identifier, a surveillance and control word of 27-bits and the 24-bit ICAO address. Each data bit takes 1 µs to transmit. A longer data block can also include an additional message field of 56-bits or an extended message field of 80-bits, depending on the format identifier. [18]

The format identifier can be one of 25 different downlink or uplink modes signifying the type of message content being transmitted. In civil aviation, only the formats in table 2.5 and 2.6 are used.

2.4.1.2 Limitations

TCAS II operates under the following constraints [19]:

1. TCAS cannot detect aircraft that are not equipped with an operational transponder.
2. The full range of coordinated RAs can only be exchanged between aircraft both equipped with Mode S. RAs cannot be issued for aircraft without an altitude reporting transponder (Mode A).
3. Only vertical resolution manoeuvres can be issued. Some situations may exist where horizontal RAs are more favourable.

Table 2.5: Civil Aviation Mode S Uplink Formats [9]

Format Identifier	**Message Content**
UF0	Short air-to-air ACAS
UF4	Surveillance altitude request
UF5	Surveillance identity request
UF11	Mode S only all-call
UF16	Long air-to-air ACAS
UF20	Comm. B altitude request
UF21	Comm. B identity request
UF24	Comm. C extended length message (ELM)

Table 2.6: Civil Aviation Mode S Downlink Formats [9]

Format Identifier	**Message Content**
DF0	Short air-to-air ACAS
DF4	Surveillance altitude reply
DF5	Surveillance identity reply
DF11	All-call reply
DF16	Long air-to-air ACAS
DF20	Comm. B altitude reply
DF21	Comm. B identity reply
DF24	Comm. D extended length message (ELM)

4. Advisories are not issued against aircraft travelling at excessive vertical rates (> 10,000 feet per minute).

5. TCAS will automatically fail if input is lost or disabled from on-board instruments such as the radio altimeter, barometric altimeter, transponder or inertial navigation system.

6. All RAs are inhibited below 1000 ft above ground level. “Descend” RAs are inhibited below 1100 ft and “Increase Descend” RAs are inhibited below 1450 ft. “Climb” and “Increase Climb” RAs can also be inhibited above certain altitudes, but this is only set when TCAS is installed or updated.

7. Due to interference, TCAS may not be able to display all transponder equipped aircraft in heavy traffic environments.

8. Terrain proximity warnings take precedence over TCAS warnings.

2.4.2 Ground Proximity Warning System

The GPWS is a mechanism that helps pilots to avoid collisions with terrain. It relies on the use of a radar altimeter to determine the aircraft's height above the ground. The system monitors trends in the terrain and warns the pilot if the aircraft is in danger of an imminent collision. Warnings include those against excessive descent rate, steep bank angle, low terrain clearance and wind shear. GPWS warnings take precedence over TCAS warnings.

A shortcoming of the GPWS is that it only detects changes in terrain directly underneath the aircraft. The GPWS could fail to predict a sudden change in terrain such as a cliff or tall building. Improvements were therefore made to form the Enhanced Ground Proximity Warning System (EGPWS). The new system has access to GIS maps and GPS satellites to help determine where an aircraft is flying relative to dangerous terrain.

Another flaw in the original GPWS is the lack of warning in short landing scenarios. The GPWS would recognise that landing gear has been deployed and would go into standby mode, ignoring any imminent collision with the ground. The EGPWS is linked to a database of the world's airports containing the exact location and descent profile of each runway. Using this information, it is able to warn the pilot if the aircraft is descending prematurely relative to the runway descent profile.

A term commonly used when referring to terrain avoidance is the Terrain Awareness and Warning Systems (TAWS). The FAA created this term to encompass all past, present and future systems that adhere to terrain avoidance objectives. The GPWS and EGPWS can therefore both be classified as TAWS.

2.4.3 Portable Collision Avoidance System

PCAS was created by Zaon Flight Systems Incorporated for light and small private aircraft operating under VFR. The system is more simplistic and passive than TCAS, but offers similar features in that it detects the range and altitude of nearby transponder-equipped aircraft. PCAS also has the ability to alert the pilot if an intruder is getting closer or further away. More modern versions of PCAS are paired with GPS to provide a visual display of intruder locations. [20]

The primary difference between TCAS and PCAS is that reaction to TCAS advisories is mandatory whereas PCAS warnings serve merely as a suggestion. PCAS also does not require special installation like TCAS and it is also less expensive. PCAS is typically equipped on aircraft using Mode A or C transponders.

2.4.4 Flight Alarms

FLARMS are small electronic devices designed to be used for collision avoidance between short-range glider aircraft. FLARMs use GPS and a barometric sensor to determine position. This information, along with a forecast of trajectory, is transmitted to other FLARM devices within a 3-5 km radius. The system can predict up to 50 collisions at a time and warns the pilot aurally. [21]

2.4.5 Air Traffic Control

ATC is a ground-based service that facilitates the flow of air traffic, provides information to aircraft and primarily enforces traffic separation rules. At an airport, ATCOs are situated in tall control towers where they have unrestricted visibility of airport airspace. ATCOs also observe aircraft location, speed and trajectory information collected from aircraft transponder interrogations and long range radar above the control tower. The ATCO must use these observations to give clearance in a sequential manner that allows smooth operation of all aircraft in this area without conflict. ATCOs communicate with the pilot via radio or by using light signals.

ATCOs have varying responsibilities and operate in different sectors at an airport. The region in which they operate around an airport is called the Terminal Control Area (TCA) in the US or the Terminal Manoeuvring Area (TMA) in Europe. The TCA or TMA consist of multiple cylindrical layers that get larger with altitude centred at the airport. ATCOs responsible for managing ground, take-off and landing operations work in a 5 mile (8 km) radius around the airport. Those responsible for coordinating approach, departure and queueing operations are active in a 5 to 50 nautical mile (9 to 92 km) radius.

2.4.6 Obstacle Collision Avoidance System

OCAS is a ground-based system that is placed near obstacles such as power lines, telecommunications towers and wind turbines, which are a threat to low-flying aircraft. OCAS uses low-power radar to detect the ground speed, heading and altitude of nearby aircraft and calculates whether or not it is appropriately clear of the obstacle.

OCAS is an obstacle-specific rule-based system. The first warning to a pilot is to turn on bright flashing lights installed on the obstacle. A second, aural warning in the cockpit is issued if the first warning is ignored. The warning is transmitted from OCAS to the cockpit via radio. [21]

2.4.7 Automatic Dependent Surveillance - Broadcast

ADS-B is a modern, cooperative system that uses ground stations, GPS satellites and an enhanced data-link to relay information in real-time between all parties involved in air traffic management. In figure 2.5, the basic operation of ADS-B can be observed. Aircraft obtain their position from GPS satellites and other information from sensors on-board. They then simultaneously broadcast this information, along with their unique ICAO 24-bit identification code, to other aircraft and ground stations. Ground stations can then relay this information to ATC towers.

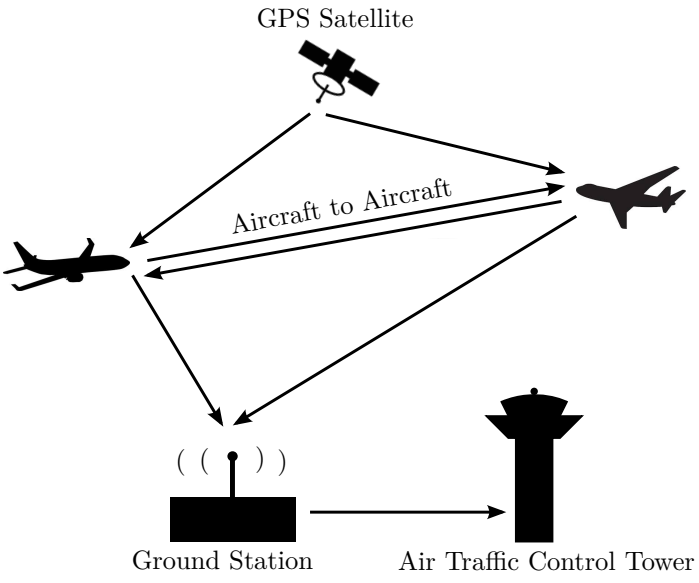


Figure 2.5: ADS-B Communication

ADS-B has the potential to replace or supplement radar-based surveillance systems. ADS-B is much more accurate than radar and allows aircraft to obtain up-to-date information about nearby vehicles or terrain that might pose a threat of collision [22]. The range of ADS-B is approximately 270 to 320 kilometres from airborne component to airborne component and 288 kilometres from airborne component to ground infrastructure [23].

ADS-B has two modes of operation: In and Out. Aircraft equipped with ADS-B In have the ability to receive information directly from other aircraft as well as from ground sta-

tions.

ADS-B Out allows an aircraft to broadcast the following data [22]:

- Horizontal position (latitude and longitude)
- Barometric altitude
- Aircraft identification (24-bit address and mode A code)
- Emergency Status
- Special Position Indicator (SPI) when selected
- Intent and weather information (depending on the transponder mode)

ADS-B Out requires that an aircraft be equipped with a Mode S transponder or Universal Access Transceiver (UAT). Both data-links are bi-directional, but Mode S works at 1090 MHz and UAT works at 978 MHz. UAT has the advantage of extra available uplink bandwidth, which means that additional weather information could also be communicated. However, UAT requires a separate radio device, while Mode S is currently incorporated into most already onboard aircraft transponders. UAT is also only presently used within the US and Mode S is used everywhere else in the world [24].

It is noted that all aircraft do not have to be capable of ADS-B In for the system to work, but only ADS-B Out. This is because all information broadcast by aircraft with ADS-B Out is received by ground stations and relayed to ATC. The ATCOs can then use this information to verbally communicate the location of surrounding aircraft to a particular aircraft that is equipped only with ADS-B Out. ADS-B Out broadcasts information once per second and therefore allows aircraft and ground stations with receivers to get an accurate measure of the current states of all aircraft in a sample airspace [23].

ADS-B In, however, provides the advantage of being able to receive that information directly and represent it on the cockpit display system with a GPS map. ADS-B In also requires the addition of a Mode S receiver, which is not already included with the Mode S transponder. However, TCAS also requires a Mode S receiver, so if an aircraft is equipped with TCAS, it is also equipped with the Mode S receiver [24].

2.5 The Future of Air Traffic Management

There are currently two projects under development within Europe and the US to modernise air traffic management by 2025: SESAR and NextGen.

2.5.1 Single European Sky ATM Research

The fragmentation of European airspace is estimated to cost €4 billion per year. This fragmentation is due to inefficient routing of flights. In 2010, the average flight to or from Europe travelled 49 km farther than its direct route. Furthermore, there is a concern for congestion in airspace, which results in passenger delays, higher airline costs and more Carbon emissions. In 2010 alone, 9.5 million flights were recorded. This is predicted to increase to 17 million by 2030.

SESAR is a collaborative research effort with the overall goal of enhancing air traffic management performance. The following are the key objectives of SESAR:

- Increase Europe's airspace capacity by 27%
- Reduce accident risk by 40% per flight hour in anticipation of increasing traffic levels
- Reduce environmental impact by 2.8% per flight
- Reduce cost by 6% per flight

This thesis aims to contribute to the reduction of accident risk. It is therefore important that the specific requirements of SESAR collision avoidance can be met. These requirements include the reduction of ATCO workload with increased automation of risk management, the use of enhanced surveillance technology, the update of altitude capture laws and the implementation of a short-range collision avoidance system specific to airport environments.

2.5.2 Next Generation Air Transportation System

Like SESAR, the aim of NextGen is to enhance air traffic communications, optimise flight routes, reduce Carbon emissions and improve safety. By 2018, NextGen is estimated to reduce passenger delays by 35%, reduce Carbon emissions by 14 million tons and save airlines $23 billion in fuel costs.

The focus of NextGen lies with the improvement of four major elements.

- Large-scale implementation of ADS-B technology will improve knowledge of aircraft intent for both air traffic controllers and pilots. The FAA plans to mandate the avionics required for ADS-B to function effectively.
- Current voice communication between aircrew and ground controllers will be supplemented by a data communication link to provide better clarity.

- The creation of a national weather information system will facilitate optimal routing of flights and the reduction of delays due to weather.
- The restructure of the voice communications network will allow for the dynamic reallocation of airspace and the optimisation of ATCO workload.

In terms of collision avoidance, NextGen aims to reduce the workload of ATCOs by automating more processes and increasing the responsibilities of airborne systems. Additional collision avoidance performance targets are also being developed specific to airport environments.

2.6 Existing Probabilistic Conflict Detection Algorithms

Early attempts at conflict prediction were performed using Monte Carlo simulation. Yang and Kuchar then improved upon the algorithm to create the Geometric Monte Carlo method. Since then, various other methods have been developed. A novel method by Van Daalen, called probability flow, is one of the most recent developments in conflict prediction and is the focus of this thesis. A discussion of this method as well as those that led to its development follow in this section.

2.6.1 Monte Carlo

Monte Carlo simulation is a method of estimating a result by sampling a known input set from a probability distribution over the domain. The simulation is run over a period of time to obtain as many samples as possible. The more random samples obtained from the input set, the more accurate the estimation will be of a particular result.

In the case of determining the probability of conflict for a host aircraft, the sampling space is the set of all possible aircraft locations in 3D Euclidean space. To reduce computational complexity, static obstacles such as terrain and protected airspace are assumed to have known locations and volumes. A probability density function (PDF) describing the uncertainty of the initial aircraft states is sampled.

Each aircraft state will have a corresponding PDF with a mean determined from on-board sensors and a standard deviation representing cross-track, along-track and altitude variability caused by disturbances. An aerodynamic model of the aircraft is then used to forward simulate a random trajectory. Each simulation is initialised with a different random sample from the PDF of the initial state vectors. The model takes the desired flight

route and associated target states as input.

Each simulation results in a random trajectory. The set of all N_{MC} random trajectories represents all possible locations the aircraft could be within the look-ahead time. Each random trajectory is checked for conflict and counted. The total number of detected conflicts divided by the total number of simulations N_{MC} is the probability of conflict.

It is therefore intuitive that as the number of simulations tends to infinity, we obtain a ground-truth accuracy on the probability of conflict. Performing an adequate number of simulations to be confident in the prediction result is very time consuming, depending on the complexity of the aircraft model. Accuracy is also dependent on how realistically we are able to model the aircraft. The Monte Carlo method therefore offers a trade-off between accuracy and efficiency.

In an airport environment where obstacles are constantly moving, obstacle motion also needs to be modelled and simulated using the Monte Carlo method. The number of aircraft that exist in the sample space is a linear scaling factor to the number of simulations required, which is proportional to the computation time. This is the largest disadvantage of the Monte Carlo method. It is therefore unrealistic to use this method for real-time conflict detection on-board aircraft. Another disadvantage of the Monte Carlo method is that if N_{MC} is not sufficiently large, the error relative to small calculated probabilities is very high. The relationship between the trajectory (aircraft position) standard deviation σ_{MC} and number of simulations N_{MC} is

$$\sigma_{\text{MC}} = \frac{1}{2\sqrt{N_{\text{MC}}}} \text{ where } \sigma_{\text{MC}} \text{ is inversely proportional to } N_{\text{MC}} \text{ [2].} \tag{2.6.1}$$

If we obtain a small probability of conflict, but N_{MC} is small, we get large uncertainty on a small result. We estimate the standard deviation on the probability of conflict P_C using equation 2.6.2.

$$\sigma_{PC(MC)} = \sqrt{\frac{1}{N_{MC}} P_C(1 - P_C)} \tag{2.6.2}$$

Monte Carlo simulation is a valuable benchmark of accuracy for other approaches when simulated with many samples over a long period of time.

2.6.2 Yang and Kuchar (Geometric Monte Carlo)

In 1998, Yang and Kuchar developed a method of reducing the number of computations required in a Monte Carlo simulation, thus decreasing running time [2]. The main simplification of their method is implemented by approximating the trajectory between major trajectory change points (TCPs) as straight lines, where TCPs represent notable course or speed transitions.

Like with normal Monte Carlo simulation, the aircraft obtains its initial state information from a random sample of the state PDF at time = 0. The velocity vector is then held constant and the remaining states are projected forward in time. The velocity state is only updated at each TCP. Yang and Kuchar simplify the method further by performing simulations in the host aircraft's frame of reference - an idea originally proposed by Paielli and Erzberger [14]. If the host's expected trajectory is subtracted from the intruder's, the trajectory of the intruder relative to the host is obtained. This allows the conflict region to be placed at the location of the reference aircraft (the origin) and the number of intersections between the conflict region and relative trajectory is accumulated. The total number of intersections recorded over N_{GMC} simulations divided by the total number of random straight-line trajectories generated is the probability of conflict.

Yang and Kuchar succeeded in reducing simulation time to an order of 1 second. Using 10,000 Geometric Monte Carlo iterations for a two-dimensional two-aircraft scenario, they achieved an error of 1.5 % in the probability of conflict. The speed and accuracy required in a real-world scenario depends mostly the nature of the scenario, type of aircraft, level of uncertainty and separation requirements. Yang and Kuchar postulated that a worst-case accuracy of 5 % would be sufficient to differentiate between conflict and no conflict [2].

2.6.3 Wangermann and Stengel

Wangermann et al. [25] refer to aircraft, ATCs and ATM systems in a given environment or system as agents. Agents are then defined as entities that can make decisions affecting the entire system based on data it has available. Wangermann et al. propose a cognitive model of decision-making and control for a so-called intelligent agent. This means that agents can have varying interests and priorities when deciding how to react to specific situations.

According to Yang et al. in 2000 [10], this method is the only probabilistic conflict avoidance algorithm that is proposed to work for global (multiple aircraft) scenarios, as opposed

to the pairwise approach in which each host and intruder pair is checked for conflict sequentially. The method defines the state of all agents in the environment and assumes that no agent can have perfect knowledge of the entire system. Principled negotiation is employed to determine a set of possible trajectories that would result in mutual gain for all agents. The process consists of four stages:

1. **Initiation**
   Agent 1 receives limited data regarding other agents and the environment. The system states and plans are initialised. This step includes improving the current plan or initialising an emergency plan.

2. **Invent options for mutual gain**
   Agent 1 comes up with a set of options that would be beneficial for all agents, particularly aircraft. This is done by using a process like Monte Carlo. The Monte Carlo trajectories are then limited using a set of rules. These options are then transmitted to Agent 2.

3. **Assessment of options against objective criteria**
   Agent 2 then receives these options and uses objective criteria to choose the optimal option for themselves based on fuel consumption or time for example. The agent either accepts an option or rejects all the options. If the agent rejects the options proposed, the agent must propose an alternative set of options.

4. **Response**
   Agent 2 sends the responses back to Agent 1. The process is continuously repeated in real-time.

Through continuous repetition, this system resembles a feedback loop in which the most optimal, mutually beneficial plan is chosen for all agents. Problems with this method may arise when consensus cannot be reached leading to the degradation of conflict alert time. The principles by which negotiation is achieved also do not incorporate uncertainty into the problem, which may result in inaccurate trajectory propagation.

The probability of conflict is never explicitly calculated in this method because scenarios of conflict are not considered valid options. Because it is a cooperative method, it means that the probability density functions of each agent's position are dependent on one other and one would need to calculate the joint probability density function of all the agents in order to calculate the probability of conflict. This would be far too computationally expensive for execution in real-time. However, this is an interesting approach to consider in comparison with pairwise approaches.

2.6.4 Jones (PDF Propagation)

A more efficient method of predicting conflict was proposed by Jones in 2006 [26]. Instead of taking a repeated outcome approach like Monte Carlo, Jones suggested the propagation of state distribution through a combination of approximation and convolution.

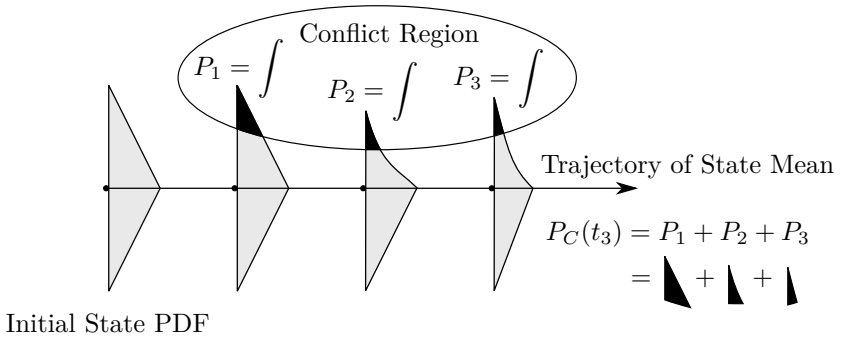


Figure 2.6: PDF Propagation and Distortion Process [4]

At each discrete time step in the future, the overlap between the state PDF and the obstacle conflict region is integrated and accumulated. The total overlap accumulated along the path at any time instant represents the total risk of conflict up to that time. Once an intersection of the state PDF is accumulated, it is removed from the PDF before it is propagated further.

This ensures that repeated accumulation of the same risk is avoided, but it also results in a distortion of the state PDF. This makes it increasingly difficult to calculate overlap and continue propagation. Jones describes the PDF propagation process as a flow of probability space into a hazard, an idea that inspired the work of Van Daalen discussed in the next section. A two-dimensional representation of this process can be seen in figure 2.6.

To simplify the complex calculations associated with propagation of the distorted PDF, Jones therefore assumes that the PDF remains unchanged, resulting in an upper bound to the probability of conflict. This means that the probability of conflict is slightly larger than the true value, leading to a more conservative estimate of conflict.

2.6.5 Van Daalen (Probability Flow)

The conflict detection method by Van Daalen is the extension of work done by Jones [4]. To the author's knowledge, probability flow and Monte Carlo are the only existing probabilistic methods capable of handling complex, non-linear, uncertain scenarios. Monte Carlo simulation cannot be performed in real-time, which is why probability flow is the primary focus of this thesis. It has the potential to most accurately perform real-time probabilistic

conflict detection in a civil aviation context.

The risk accumulation method described by Jones is computationally expensive and time consuming because it requires volume integration. Jones and Van Daalen therefore later proposed a solution by defining probability flow as the accumulation of the rate of probability increase at the boundary of the obstacle conflict region – rather than the accumulation of the overlap between the aircraft state PDF and the interior of the obstacle conflict region. The volume integration previously required is thereby reduced to surface integration, drastically decreasing computation time.

A proof by Van Daalen [13] shows that the state PDF can be assumed unchanged with each encounter with the conflict region. However, a tight upper bound is still present. The difference to the true probability of conflict is sufficiently small to avoid false alarms and sufficiently large to provide a conservative safety margin. This makes the algorithm ideal for application in aviation where safety is prioritised above other performance factors.

2.7 Case Studies

By studying the records of previous aircraft collisions, insightful simulation scenarios can be selected for the testing of conflict detection algorithms. According to the Bureau of Aircraft Accidents Archives (B3A), a total of 1581 crashes have occurred since the beginning of 2005. The aircraft involved in these accidents include those belonging to military authorities, commercial airlines and private aviation companies [27]. It is important to note that although the focus of this thesis is specific to commercial aviation, the concepts explored herein can be applied to all types of airborne vehicles.

Significant commercial aircraft collisions can be categorised into three different types: controlled flight into terrain (CFIT), uncontrolled flight into terrain (UFIT) and mid-air collision.

2.7.1 Controlled Flight into Terrain (CFIT)

CFIT occurs when an aircraft, which is considered to be airworthy and under full control of a pilot is unintentionally flown into a stationary obstacle, such as the ground, a mountain, building or mass of water. The primary cause of CFIT accidents is due to a pilot's loss of spatial awareness or the disregard of EGPWS warnings and instrument readings. A few accidents have, however, been attributed to malfunctioning instruments.

2.7.1.1 Poor Visibility

On 19 April 2000, Air Philippines Flight 541 was denied clearance from ATC for landing because of runway traffic. The aircraft proceeded to circle the airport in anticipation of a landing from the opposite direction. Visibility was poor due to low-lying cloud cover and the pilot descended below the required 1500 ft at 8 km away from the airport. The aircraft crashed into a coconut plantation on an island 500 ft above sea level. It is unclear whether or not the pilot received EGPWS warnings, but the accident was attributed to pilot error.

A more recent incident of pilot error occurred on the local Airblue Flight 202 on 28 July 2010 in Pakistan. The aircraft was circling the Benazir Bhutto airport for landing when it crashed into the Margalla Hills near Islamabad. Prior to the crash, the pilot was instructed by ATC to stay within a 9.3 km radius of the airport, but failed to do so, veering 15 km North of the airport. Heavy rains and fog compromised visibility and caused ATC to lose contact with the aircraft after issuing the instruction. The cockpit voice recorder captured evidence that multiple "terrain ahead" warnings were issued by the EGPWS starting 40 seconds before the crash, but evasive action was not taken in time.

2.7.1.2 Short Landing

A short landing is when an aircraft is accidentally piloted into the static environment during an approach or landing procedure before reaching the runway. An example of a short landing incident occurred in July 2013 when Asiana Airlines Flight 214 hit the sea wall dividing runway 28L at San Francisco international airport and the Pacific Ocean.

The pilot was performing a visual approach under VFR assisted by the runway's Precision Approach Path Indicator (PAPI). A PAPI is a light array that shines different colours depending on the angle at which they are observed. If the pilot sees more white lights than red, the aircraft is descending above the recommended glide-slope. Vice versa applies if more red lights are observed than white. The optimal glide-slope is achieved if the pilot can see an even PAPI light ratio.

After the crash, the National Transportation and Safety Board (NTSB) of the US found that Asiana Airlines Flight 214 descended well below the desired glide-path, causing the tail of the aircraft to strike the sea wall. The pilot reportedly tried to abort the landing, but was unable to do so in time.

2.7.1.3 Instrument Malfunction

Instances of instrument malfunction are extremely difficult to address. A malfunction creates mistrust in the equipment, which can instigate incorrect flight management. In January 2000, Kenya Airways Flight 431 crashed into the sea off the coast of Côte d'Ivoire shortly after take-off. An electrical fault caused a false stall warning, prompting the pilot to start descending. A GPWS alert was not issued because the stall warning took precedence. Only after an over speed warning was issued, did the pilot initiate a climb, but it was too late.

2.7.2 Uncontrolled Flight into Terrain (UFIT)

UFIT accidents occur when a pilot can no longer control an unstable aircraft because of damage or malfunction. A collision with terrain in this case is often unavoidable even if the correct procedures are followed by the flight crew.

Two examples of UFIT are Sita Air Flight 601 in September 2012 and Dana Air Flight 992 in June 2012. The former crashed on the banks of a river in Nepal after the pilot reported technical issues following a suspected bird strike. The latter crashed upon airport approach in Lagos, Nigeria after the pilot reported dual engine failure.

2.7.3 Mid-air Collision

A mid-air collision occurs between two or more aircraft and is less common than CFIT [28]. Because aircraft travel at such high speeds, a mid-air collision is usually accompanied by severe damage to at least one aircraft followed by UFIT. Mid-air collisions are more challenging to predict and avoid because all parties involved are independently controlled.

One of the most well-known cases of mid-air collision occurred in Überlingen, Germany between Bashkirian Airlines Flight 2937 and DHL Flight 611. Both aircraft were flying at an altitude of 36,000 ft. Only 30 seconds before the crash, the air traffic controller on duty noticed that they were on a collision course. He instructed the Bashkirian Airlines flight to descend by 1000 ft and returned his attention to another aircraft landing at a nearby airport. Meanwhile, the DHL flight received a TCAS RA to descend and the Bashkirian Airlines flight received a corresponding RA to climb. The passenger aircraft ignored the TCAS advisory and listened to the ATC instruction instead. The cargo plane followed the TCAS advisory leading both aircraft on a descent towards one another.

The DHL aircraft's vertical stabiliser sliced through the fuselage of Flight 2937. Both aircraft broke up and crashed in the fields below leaving no survivors. The crash prompted ICAO to issue new policy stating that TCAS RAs should be given preference over ATC instructions. The management of air traffic control services was also reviewed.

2.7.4 Observations

From these case studies, it is clear that the safe operation of passenger aircraft is reliant on many factors. Firstly, training pilots to monitor their instruments closely and to heed EGPWS warnings immediately is crucial in preventing CFIT accidents. The improvement of collision avoidance systems is without purpose if pilots cannot trust and operate such systems reliably.

A collision avoidance system is susceptible to the same risk of malfunction as any other flight instrument and is only reliable if checked and maintained regularly. It should also be noted that perhaps cockpit warnings should not be prioritised based on the type of risk, but rather the urgency of the risk.

Furthermore, a collision avoidance system will unfortunately not help to prevent UFIT accidents. The only preventative measures for this type of accident is the rigorous testing of all aircraft components before they are deployed paired with strict, thorough maintenance routines. Lastly, correct legislation and policy governing air traffic management as well as instrument use is necessary to avoid confusion and facilitate the safe operation of aircraft.

This section concludes the literature review chapter. The reader should now have a greater understanding of the project context and objectives as well as a sufficient knowledge of collision avoidance to continue with the modelling chapter.

Chapter 3

Modelling

In this chapter, the proposed conflict avoidance framework is introduced and techniques for modelling the static and dynamic environments are discussed.

3.1 Proposed Framework

A framework describes the relationship between different components in a system. A high-level framework of the proposed conflict avoidance system is shown in figure 3.1 on the following page.

The framework consists of 7 different components, referred to here as modules.

1. **Initialise**

   When an aircraft is preparing for take off, the conflict avoidance system, flight computers and other avionics are booted. The conflict avoidance system will download any information it requires preflight and initialise all its variables. This includes the GIS terrain database, aerodynamic constants, time parameters, error thresholds, flight rules, airport descent profiles and the planned flight route. All on-board sensors, instruments and control surfaces are also thoroughly checked by the pilot.

2. **Model the Environment**

   Once the conflict avoidance system has been initialised, it begins to accumulate a situational awareness of obstacles near the aircraft. This is achieved through the use of the ADS-B infrastructure discussed in section 2.4.7 on page 20. A model of the host aircraft is constructed by collecting state information from on-board sensors and instruments. The terrain and intruder Minkowski sums (discussed later in section 3.4 on page 38) are then computed.

3. **Predict the Future States**

   The conflict avoidance system should ideally prevent all conflict scenarios both on the ground and in the air. The system should therefore become active as soon as the

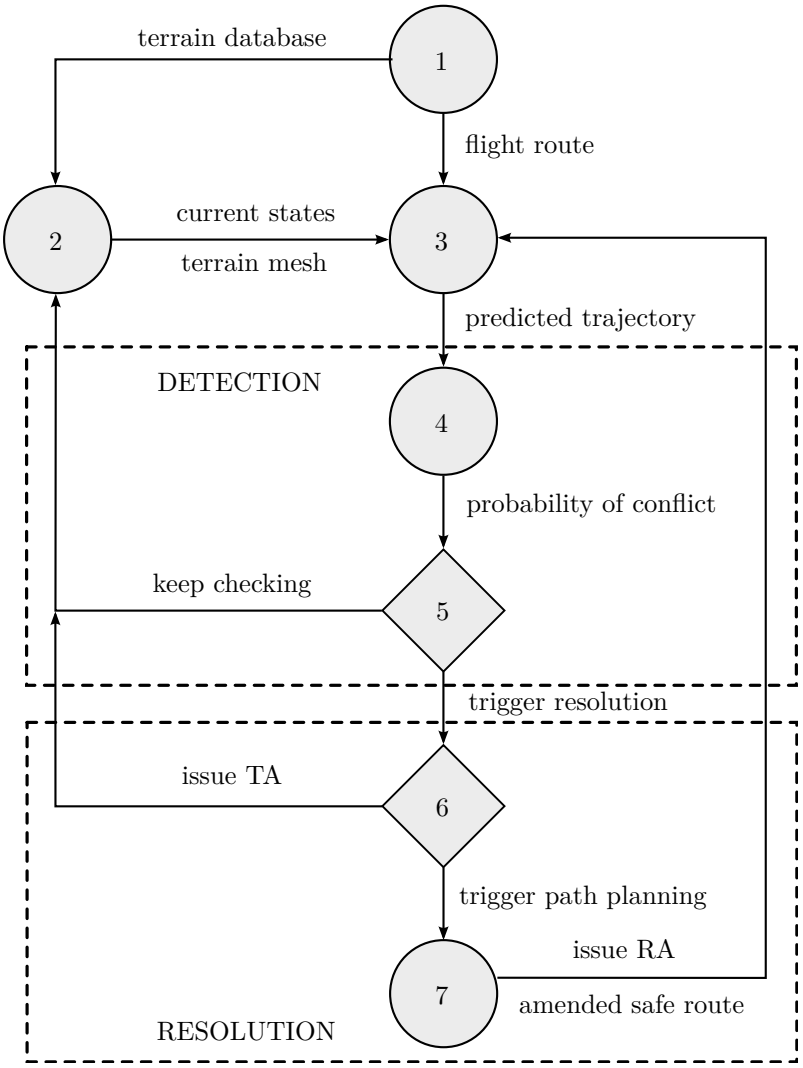


Figure 3.1: High-Level Conflict Avoidance Framework

aircraft begins to taxi. At this point, the simulation module will immediately begin to predict the aircraft's states and most importantly, its trajectory, for a specified time interval into the future. State prediction involves propagating the current states of the aircraft into the future within an envelope of uncertainty while taking available intent information into account. This is a complex task, which demands its own dedicated research focus. In this thesis, the future states are therefore assumed to resemble the given intent information within an envelope of normally distributed uncertainty. In other words, the predicted trajectory will roughly track the planned flight route. In practice, path planning algorithms should be used to determine the predicted trajectory. Currently, the aircraft is assumed to follow a straight-line in the direction of the current velocity vector.

4. **Calculate the Probability of Conflict**

The conflict detection module uses the current and predicted aircraft states to de-

termine the host aircraft's conflict probability for the specified future time interval. The probability flow method by Van Daalen is the chosen algorithm for this purpose because it is believed to offer an accurate real-time solution while producing a conservative result, making it ideal for civil aviation application.

5. **Assess the Risk**

   Once the probability of conflict has been determined, it must be compared against a predefined risk threshold. If the risk is above the threshold, a resolution process is triggered. If the risk is below the threshold, there is no perceived threat of collision and no action is taken. In civil aviation, risk thresholds are typically very low. In this thesis, a risk threshold of 5% is chosen for demonstration purposes, but the risk threshold should ultimately be chosen at the discretion of the system designer. Once the risk has been assessed, the conflict avoidance system increments the future time interval and returns to the beginning of the conflict detection process i.e. the modelling module.

6. **Decide on a Course of Action**

   If the conflict resolution module is triggered, the threat of collision is deemed dangerous and warrants corrective action. Depending on the severity and nature of the threat, a course of action appropriate to the situation is chosen. If a collision is not imminent, but the risk is still significant, the pilot is simply issued a TA. The pilot should then try to visually acquire the threat and use discretion to prevent further provocation of risk. If a collision is imminent, the path planning module is triggered.

7. **Find a Safe Alternative Path**

   Path planning methods are used to determine a safe alternative route when a high probability of conflict is detected on the current predicted trajectory. Paths which result in the smallest cost are desirable. Cost can be based on flight considerations such as fuel consumption, passenger comfort, distance or time. Potential paths are checked for conflict with a separate instance of the conflict detection module before being deemed safe. Once the optimal safe path is found, the resolution module suggests a sequence of appropriate resolution manoeuvres to the pilot. The flight route is then amended and updated in the simulation module.

This thesis presents an implementation of the conflict detection module where all other modules have been created purely to provide proof of concept for the probability flow method. All other modules are therefore not implemented as they would be in practice, but provide an adequate testing platform for the conflict detection module.

3.2 Conflict Region

As previously stated, the conflict region is a volume of empty airspace encapsulating an aircraft, which must be maintained in pursuance of regulated safety objectives. Because safe separation criteria vary for different phases of flight and classes of aircraft, the conflict region cannot be represented as a constant model.

At high altitudes, aircraft travel at high speeds. A large conflict region is therefore necessary to ensure that conflict is detected further away, allowing the pilot sufficient time to execute an evasive manoeuvre. A large conflict region would result in a conservative conflict detection system, but this could lead to a high frequency of unwarranted detections in cluttered environments. As a consequence, pilots may mistrust the system and ignore warnings, even when there may be a legitimate risk of collision.

Contrarily, in airport environments, a small conflict region is desirable where aircraft travel at low speeds and manoeuvring space is limited. The size of the conflict region is therefore kept variable throughout this thesis, depending on the simulation scenario.

The shape of the conflict region is often assumed to be a cylinder [14; 2; 29; 15] with 5 NM radius and 2000 ft height (see figure 3.2), corresponding to the current ICAO en-route separation requirements [11]. Propagating a complex shape, such as a cylinder, forward in time along a predicted trajectory is difficult because future changes in orientation cannot be anticipated, adding to uncertainty in prediction.

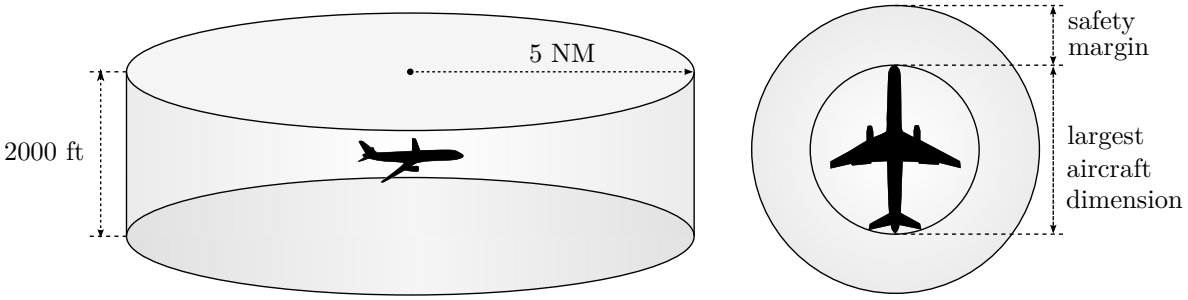


Figure 3.2: Cylindrical and Spherical Conflict Region Models

To keep uncertainty as low as possible, the conflict region is therefore assumed to be a sphere, which is impervious to changes in pitch, roll and yaw. Furthermore, because this thesis focuses largely on the airport environment, the radius of the conflict region is kept tight with the largest dimension of the host aircraft under consideration. An additional

safety margin can be added for different phases of flight at the discretion of the system designer.

3.3 Terrain

In section 2.3.3 on page 13, the concept of GIS elevation maps is introduced. Unfortunately, GIS maps are expensive and difficult to acquire for terrain near airports without appropriate permissions. The Google elevation application programmers interface (API) was identified as a viable alternative testing solution.

Google provides a free service to developers who wish to query the elevation of exact locations on the Earth. If an exact location's elevation is not known, the elevation API returns an interpolated average elevation from the 4 closest locations.

The limitations of the API under a non-commercial license are as follows:

- 2500 requests per 24-hour period
- 512 locations per request
- 25,000 locations in total per 24-hour period
- 10 requests per second

The elevation dataset is usually stored as a 2D position grid with an altitude corresponding to each position in the matrix. Having discrete data means that the terrain can only be approximated by interpolating between elevation points.

Detecting conflict with terrain involves determining if intersections occur between a predicted trajectory and the modelled surface. Calculating intersections with an arbitrary surface is a complex task because the surface might contain trends and edges that cannot be described mathematically. It is therefore common practice to break up the surface into a polygon mesh of triangles. This technique is popular for its use in computer game graphics, animation, robotics and mathematical modelling.

A triangular mesh is described by a collection of vertices, edges and faces. A vertex is a point in space that forms the corner to at least one triangle and the intersection between multiple triangles. An edge is simply a connection between two vertices. A face is an enclosed triangle described by the connection of three edges.

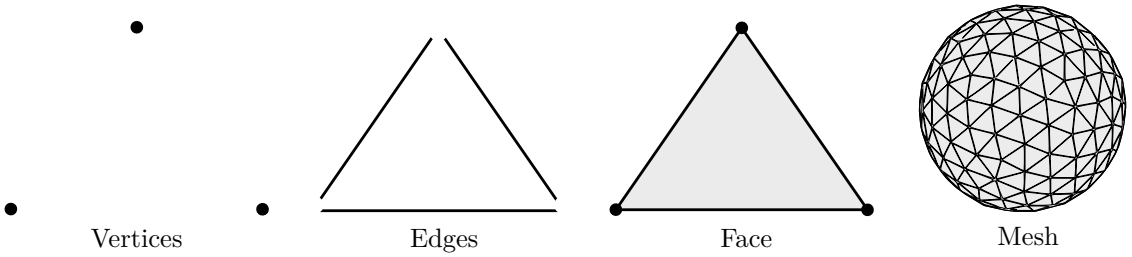


Figure 3.3: Polygon Mesh Terminology

Triangulation algorithms can be categorised into three different categories, namely Delaunay-based, region growing and implicit surface reconstruction [30]. Delaunay-based triangulation reconstructs a surface such that no point in the dataset lies within the circumcircle of any triangle in the mesh. The circumcircle (or circumscribed circle) is a circle that encloses and passes through all the vertices of a mesh face. This ensures that the minimum interior angle of all triangles is maximised. There are numerous implementations of the Delaunay method [31; 32; 33; 34; 35], but one of the most useful is contained in the open-source C++ Computational Geometry Algorithms Library (CGAL) [36].

An advantage of using Delaunay triangulation is that data points can be processed in any order, which means that the mesh can be edited once constructed. The algorithm also allows for the concentration of smaller triangles in areas of high resolution and larger triangles in areas with less gradients. Furthermore, the exact location dataset values are used, making it convenient for approximation of location-specific surfaces such as mountains, buildings and roads etc.

Although Delaunay triangulation may be very accurate, it is, however, very slow to compute due to its complexity. For large meshes in computer memory, arbitrary access patterns can lead to memory thrash [37]. Thrashing occurs when a computer's virtual memory is in a cycle of rapidly exchanging data in temporary memory for data on disk, seizing processing power needed by other applications.

The region growing method initially prunes a single triangle face from the dataset and subsequently iterates through the rest of the data in order to attach new triangles to the region's boundary. Various examples of the region growing method can be found in [38; 39; 40; 41]. This method of triangulation is extremely fast in comparison to Delaunay, but it depends heavily on user-defined parameters, sample density and data noise, making it difficult to guarantee a closed manifold model i.e. no missing edges [30].

The implicit surface method relies on the creation of a 3D signed distance function from

a set of sample points. A signed distance function is used to determine whether or not a given point lies within the boundary of a set of points. The function assigns positive values for points inside the boundary, negative values for those that lie outside and zero to points on the boundary. The surface mesh is therefore constructed from the subset of values for which the signed distance function is zero. Adaptations of the implicit surface method can be found in [42; 43; 44; 45] and a review of these methods can be found in [46]. Unlike Delaunay and region growing triangulation, the implicit surface method approximates the sample points instead of interpolating them, meaning that accuracy is heavily compromised.

For the purpose of this thesis, Delaunay-based methods are most ideal because terrain model accuracy can be preserved. Slow execution is of little consequence if terrain meshing can occur once before each flight. This is, however, only possible if there is prior knowledge of the aircraft's flight path and potential surface span. Even so, it may not be possible to mesh and store such large expanses of terrain data on flight computers.

In practice, it is suggested that small portions of anticipated terrain data be downloaded via satellite a few seconds before the conflict detection system requires it. This would allow sufficient time for meshing to occur before the terrain model is needed. In this thesis, knowledge of the predicted trajectory is assumed, so the expanse of terrain it covers is therefore meshed using Delaunay triangulation once before each simulation ("pre-flight").

3.4 Minkowski Addition

Minkowski addition is often used in robot motion planning to determine a vehicle's configuration space i.e. the available manoeuvring region in the environment. For two sets of position vectors describing the conflict regions of the host and intruder aircraft, R_h and R_i respectively, in 3D Euclidean space, the Minkowski sum can be calculated by adding each coordinate vector $\mathbf{r_h}$ in R_h to each coordinate vector $\mathbf{r_i}$ in R_i (see equation 3.4.1). This is in essence the convolution between the two aircraft's conflict regions. It can also be thought of as the union of all translations of R_h by a 3D point in R_i. [47]

$$R_h + R_i = \{\mathbf{r_h} + \mathbf{r_i} | \mathbf{r_h} \in R_h, \mathbf{r_i} \in R_i\} \tag{3.4.1}$$

To find the configuration space in an airport environment, Minkowski addition must be performed not only between the host aircraft and all intruder aircraft, but between the host aircraft and all obstacles, including terrain and protected airspace. Minkowski addition results in a "swollen" environment model allowing the host aircraft to be modelled and propagated as a point mass.

For complex shapes, Minkowski addition is a computationally expensive process. For the case of determining the Minkowski sum between the host and intruder aircraft, the choice of a spherical conflict region simplifies the problem considerably. For a non-spherical region, the Minkowski sum would need to be recomputed at each time step of propagation. However, because a sphere does not change with orientation, the Minkowski sum need only be calculated once before each flight. The Minkowski sum of the host aircraft with an intruder is then simply a sphere with a radius equal to the sum of both aircraft's conflict region radii (see figure 3.4).

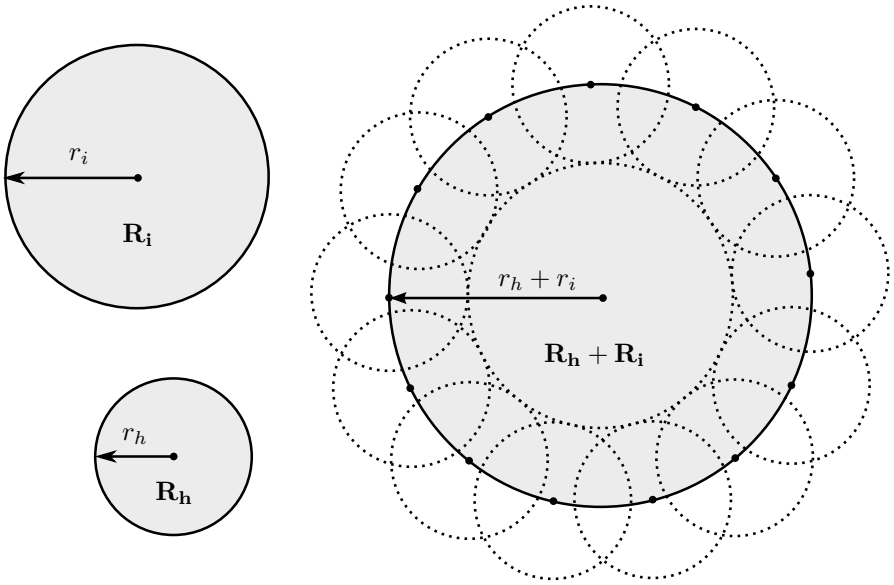


Figure 3.4: Minkowski Addition of the Host Aircraft with an Intruder Aircraft

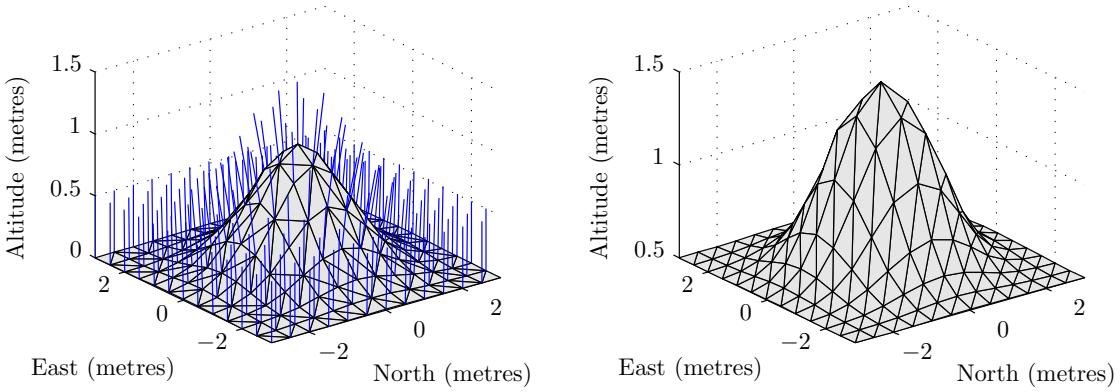
Calculating the Minkowski sum of the host aircraft with terrain is more difficult because terrain is so irregular. However, once the surface is broken up into a polygon mesh of triangles, an estimate of the Minkowski sum can be obtained. The approximate Minkowski sum is found by raising each mesh vertex by the radius of the host aircraft conflict region. This must be done in the direction of the net normal vector of the surrounding faces. For a single triangular face with vertices $\mathbf{p_0}$, $\mathbf{p_1}$ and $\mathbf{p_2}$, the normal vector extending outwards from the centre of the triangle is

$$\mathbf{n_f} = (\mathbf{p_1} - \mathbf{p_0}) \times (\mathbf{p_2} - \mathbf{p_0}). \tag{3.4.2}$$

The magnitude of each normal vector is proportional to the area of the mesh face. The net normal vector at a vertex is found by combining the normal vectors of the mesh faces sharing that vertex as follows:

$$\mathbf{n_{p_j}} = \sum_{i=1}^{N_F} \mathbf{n_{f_i}} \tag{3.4.3}$$

where N_F is the total number of mesh faces and $j \in [0, 1, 2 \ldots (N_V - 1)]$ where N_V is the total number of mesh vertices.



(a) Normal Vectors on Mesh Vertices (b) Completed Minkowski Sum

Figure 3.5: Small-Scale Minkowski Addition Example of Sphere with Terrain

The net normal vector is then scaled by its reciprocal length to get the unit normal vector at the vertex. The radius of the intruder aircraft can now be added to each mesh vertex in the direction of the unit normal vector to find the approximate Minkowski sum.

$$\mathbf{p_{new}} = \mathbf{p_{old}} + r_h \mathbf{n_p} \tag{3.4.4}$$

An example of minkowski addition between a small sphere of radius 0.5 metres with a meshed 3×3 metre hill is shown in figure 3.5.

3.5 Generic Aircraft Model

A model of the host aircraft's dynamics is needed to perform Monte Carlo simulation. In this section, the axis systems, notation, sub-systems and basic dynamic relationships associated with such a model are introduced. The specifics of the model used for simulation in this thesis is discussed at length in section 4.3 on page 48.

3.5.1 Principal Axis System

The principal axis system is fixed to the aircraft with the origin placed at the centre of mass C_m (see figure 3.6 on the following page). The longitudinal axis X_P lies in the plane of symmetry extending through the aircraft's nose and tail, the lateral axis Y_P extends perpendicularly to the fuselage through both wings and the normal axis Z_P extends through the top and bottom of the aircraft. By convention, the positive longitudinal direction is towards the nose, the positive lateral direction is towards the right wing and the positive

normal direction is downwards. Rotation about the longitudinal, lateral and normal axes is called roll, pitch and yaw respectively. [5] In the principal axis system, the notation described in table 3.1 is used.

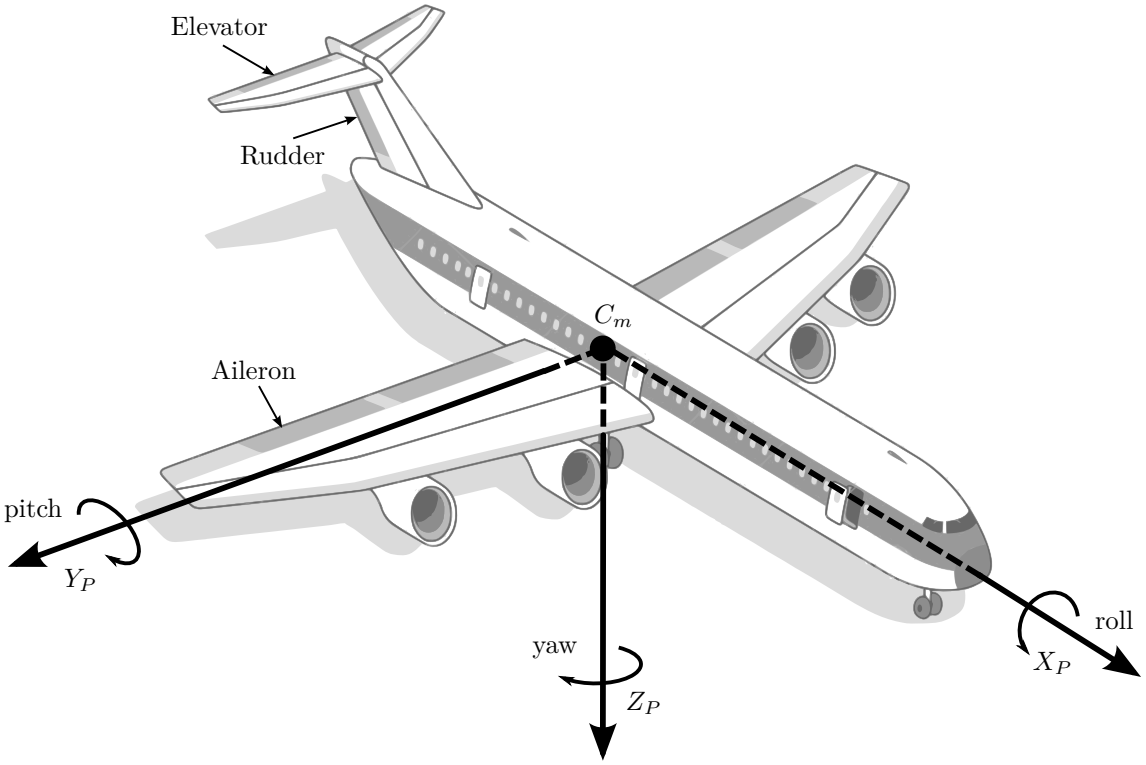


Figure 3.6: Aircraft Principal Axis System

Table 3.1: Aircraft Principal Axis Symbol Definitions

Symbols	**Meaning**
X, Y, Z	Force vector coordinates in the X_P, Y_P and Z_P axes respectively
L, M, N	Roll, pitch and yaw moments respectively
U, V, W	Linear velocity vector coordinates in the X_P, Y_P and Z_P axes respectively
P, Q, R	Roll, pitch and yaw rates (angular velocity) respectively
$\overline{V}$	Velocity magnitude calculated by: $\overline{V} = \sqrt{U^2 + V^2 + W^2}$
α	Angle of attack between the $X_P Y_P$-plane and $\overline{V}$ calculated by: $\alpha = \arctan\left(\frac{W}{U}\right)$
β	Angle of sideslip between the $X_P Z_P$-plane and $\overline{V}$ calculated by: $\beta = \arcsin\left(\frac{V}{\overline{V}}\right)$

3.5.2 Inertial Axis System

In order to apply Newton's first law of motion, the use of an inertial reference frame is necessary. Any reference frame that is in uniform motion relative to the stars preserves the validity of Newton's first law. In aerospace, the North-East-Down (NED) axis system is used (shown in figure 3.7). The East-West axis is tangent to the Earth's circles of latitude, or parallels, and the North-South axis is tangent to the Earth's lines of longitude, or meridians. Assuming the Earth to be a sphere, the Up-Down axis extends in the direction of the Earth's centre.

In aviation, the origin of the axis system is usually fixed to the aircraft's centre of gravity. However, for short-range applications, the axis system can be fixed to a flight starting point, but this assumes a flat, non-rotating model of the Earth. [5] This assumption is justified for use in this thesis because the look-ahead time in the presented scenarios is small (30 - 60 seconds), resulting in short travelled distances.

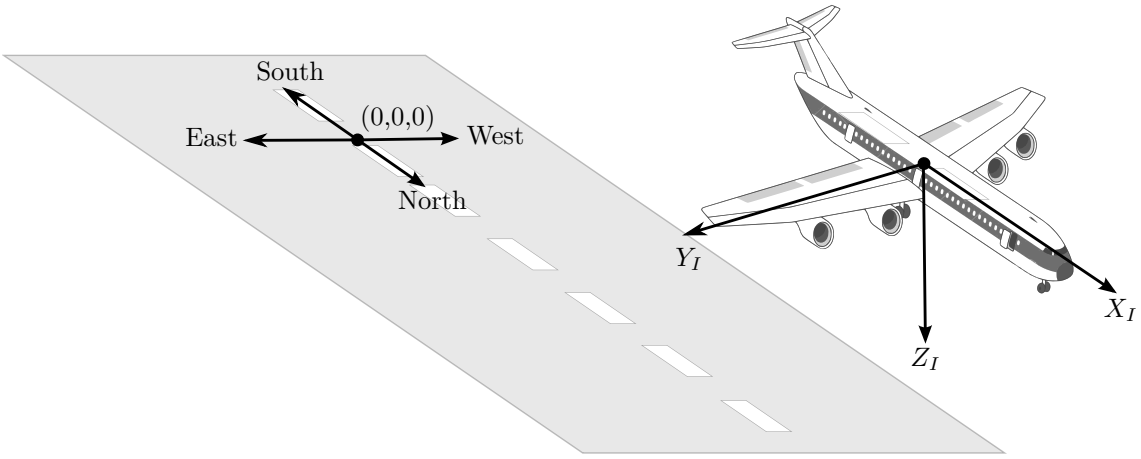


Figure 3.7: NED Axis System

In the inertial axis system, the notation in table 3.2 is used. Euler angles Φ, Θ, Ψ are popular for describing roll, pitch and yaw angles because of their simplicity. However, a singularity always exists at a $\pm$ 90 degree pitch angle for Euler angles ordered 3-2-1.

Table 3.2: Aircraft Inertial Axis Symbol Definitions

Symbols	Meaning
N, E, D	Position vector coordinates in the North, East and Down axes respectively
Φ, Θ, Ψ	Euler 3-2-1 attitude parameters of the principal axis system relative to the inertial axis system

The order describes the sequence of rotation following the initial alignment of the principal and inertial axis systems. In the case of 3-2-1, the sequence is first yaw, then pitch and lastly roll (see figure 3.8). To avoid the singularity present when using Euler angles, alternative parametrisations such as Quaternions can be employed, but these are more complex to work with and less intuitive than Euler. Fortunately, on conventional commercial passenger flights, a 90 degree pitch angle is never achieved and the singularity is never realised.

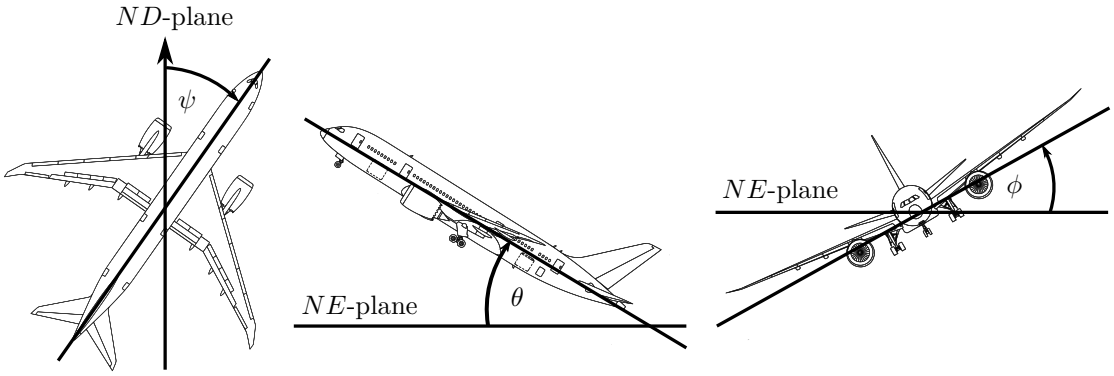


Figure 3.8: Euler Angles of Attitude

3.5.3 Conversion between Axis Systems

For the two axis systems described above, it is sometimes convenient to convert vectors, such as position and velocity, from the one axis system to the other. This is achieved through the use of a transformation matrix. The direction cosine matrix (DCM) in equation 3.5.1, denoted by $\mathbf{\Lambda}$, is valid only for right-hand systems.

$$\mathbf{\Lambda} = \begin{bmatrix} \cos\theta\cos\psi & \sin\phi\sin\theta\cos\psi - \cos\phi\sin\psi & \sin\phi\sin\psi + \cos\phi\sin\theta\cos\psi \\ \cos\theta\sin\psi & \cos\phi\cos\psi + \sin\phi\sin\theta\sin\psi & \cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi \\ -\sin\theta & \sin\phi\cos\theta & \cos\phi\cos\theta \end{bmatrix} \tag{3.5.1}$$

The transformation can be performed from inertial to principal coordinates as follows:

$$\begin{bmatrix} x_I \\ y_I \\ z_I \end{bmatrix} = \mathbf{\Lambda} \begin{bmatrix} x_P \\ y_P \\ z_P \end{bmatrix}. \tag{3.5.2}$$

To transform from the principal to inertial axis system, the inverse of the transformation matrix is used, which due to its orthogonality, is simply the transpose.

$$\begin{bmatrix} x_P \\ y_P \\ z_P \end{bmatrix} = \mathbf{\Lambda}^{-1} \begin{bmatrix} x_I \\ y_I \\ z_I \end{bmatrix} = \mathbf{\Lambda}^{T} \begin{bmatrix} x_I \\ y_I \\ z_I \end{bmatrix} \tag{3.5.3}$$

3.5.4 Basic Framework

A simple framework depicting the component relationships in a generic aircraft model is shown in figure 3.9. The aerodynamic forces and moments are responsible for the majority of uncertainty in the model. Various aerodynamic coefficients, describing lift and drag for instance, form part of equations that govern a specific aircraft's motion. The thrust model describes the aircraft engine propulsion capabilities and the gravitational model describes the force of the Earth's pull on an aircraft.

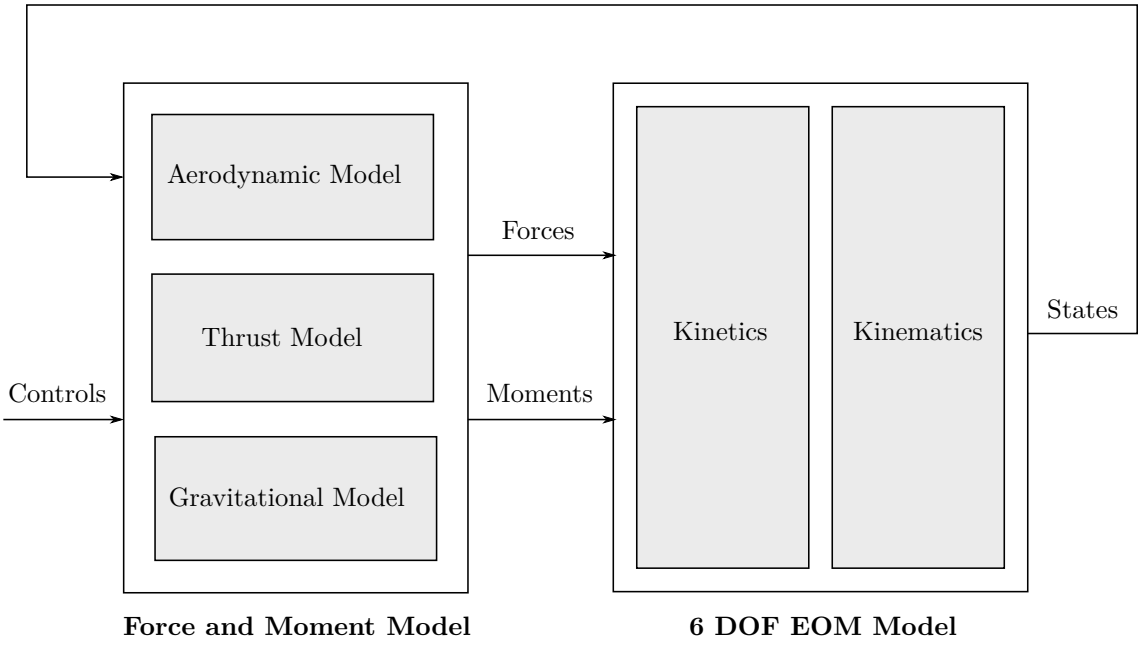


Figure 3.9: Basic Aircraft Model by Peddle [5]

Kinematics is a field of mechanics that describes the relationship between different variables of motion over time, such as velocity, attitude and position. Kinetics then relates the forces and moments acting on the aircraft with the associated kinematic states. Together, kinetics and kinematics form the 6 degree-of-freedom (DOF) equations of motion (EOM) model. [5]

Chapter 4

Implementation

This chapter discusses all steps taken in the implementation and testing of the probability flow algorithm. This includes the development of a software testing platform, the generation of terrain data for the environment model and the augmentation of the aircraft model to accurately imitate a real aircraft in Monte Carlo simulation. The definition of probability flow is given in this chapter and numerical integration techniques are examined for approximating the complex integrals present in the definition. Lastly, a discussion on how to interpret a combined probability of conflict for multiple conflict events is presented.

4.1 Software Overview

Before examining the technical details associated with the Monte Carlo and probability flow methods, the implemented software architecture is introduced. The architecture is similar to the proposed framework in figure 3.1 on page 33, but the flow of execution more closely resembles a sequential set of steps.

1. **Variable Initialisation**
   The first step in the simulation of each scenario is to initialise all time constants and sample specifications. The paths to all necessary libraries and data files are also created.

2. **Aircraft and Terrain Data Retrieval**
   The next step is to define the specifications of all aircraft involved in the scenario including their dimensions, class and initial state values. These initial state values along with any intent information is retrieved from separate data files to simulate the procedure of a conflict avoidance system fetching information from external sources, such as ADS-B and sensors for instance. Terrain data is also retrieved from an external data file as if to simulate a GPS or EGPWS query, but if the data for a specific location is not available, it is generated using the Google Elevation API.

3. **Conflict Region Construction (Minkowski)**
   Now that the system has collected knowledge of all aircraft and the environment,

the conflict region is constructed. This is essentially the Minkowski sum of the host aircraft with all other aircraft and the Minkowski sum of the host aircraft with the terrain elevation dataset. These regions are then meshed and stored in memory.

4. **Trajectory Prediction**
   Using the initial state information collected in step 2, the trajectories of all aircraft in the environment are predicted. If intent information is available, a set of waypoints is constructed to form the desired flight route. If no intent information is retrieved for a particular aircraft, the predicted trajectory is approximated by projecting a straight line in the direction of the initial velocity vector. This method of trajectory prediction may seem primitive, but it is not the primary focus of this thesis and is therefore only presented to provide proof of concept.

5. **Monte Carlo Simulation**
   For each aircraft in the environment, N_{MC} Monte Carlo forward simulations must be performed. The aircraft model is initialised with the time constants specified in step 1 and the waypoints generated in step 4. The initial state values of the model are randomly selected from a pre-determined PDF of those states (discussed later in section 4.4.1 on page 57). The velocity and position along each randomly generated Monte Carlo trajectory is stored for each aircraft simulation.

6. **Calculation of Mean and Covariance**
   Now that N_{MC} trajectories have been generated for each aircraft, the mean and covariance along each aircraft's simulated path can be calculated. These values are required for mesh reduction in the next step as well as for the execution of the probability flow algorithm.

7. **Mesh Reduction**
   Before calculating the host aircraft's probability of conflict, mesh reduction techniques are applied to reduce computation time. Mesh reduction entails discarding all points that are not within 10 standard deviations of the mean and reducing the number of mesh faces in areas of low detail.

8. **Calculate Probability of Conflict (Monte Carlo)**
   The "ground-truth" probability of conflict is now calculated for the host aircraft. This is achieved by running a mesh intersection routine (described in section 4.4.2 on page 58) on all the host aircraft's Monte Carlo trajectories. The probability of conflict is determined by dividing the total number of detected mesh intersections by the total number of trajectories.

9. **Calculate Probability of Conflict (Probability Flow)**
   Finally, the probability flow method can be evaluated. The specifics of the algorithm are discussed in section 4.5.

4.2 Environment Model

To obtain the elevation of a particular point on the Earth, the Google Elevation API must be queried with a specific value of latitude and longitude. A query is performed using an HTTP request URL. This URL is of the form:

```
http://maps.googleapis.com/maps/api/elevation/outputFormat?parameters
```

The output is either returned in the JSON or XML description format. In this thesis, it was decided to retrieve elevation data in XML format because it is simple to understand, parse and interface with MATLAB. To make the solution simpler, an open-source XML toolbox available on MATLAB File Exchange is used to facilitate the execution and interpretation of API queries [48].

To query the elevation of a single point on Earth, the `parameters` argument in the request URL becomes `locations=latitude,longitude`. However, if the elevation along a path is desired, this becomes

```
locations=lat_start,lon_start|lat_end,lon_end&samples=num_samples.
```

By specifying the `samples` argument, discrete, equally-spaced elevation data can be obtained along a path with a specified starting and ending latitude and longitude using only a single request. A square elevation map can therefore be generated by performing N_R requests with each request containing N_R samples.

Due to the daily usage limits (mentioned in section 3.3 on page 36), an elevation map containing a maximum of 25,000 data points can be generated per day. This is equivalent to a 158×158 point map.

In order to create maps large enough to include hazardous features such as mountains, it was decided to compromise on resolution and request elevations at locations further apart than at a distance that would result in the highest resolution. The distance between samples is therefore chosen to be $0.003°$ in latitude and longitude, which is approximately 45 metres. The conversion between degrees of latitude and longitude to distance is performed using the Haversine formula:

$$\Delta M = 2r_E \arcsin\left(\sqrt{\sin^2\left(\frac{\varphi_2 - \varphi_1}{2}\right) + \cos(\varphi_1)\cos(\varphi_2)\sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right)}\right) \tag{4.2.1}$$

where r_E is the radius of the Earth, assumed to be spherical, (φ_1, λ_1) is respectively the latitude and longitude of point 1, (φ_2, λ_2) is respectively the latitude and longitude of point 2 and ΔM is the distance between the two points.

4.3 Aircraft Model

Accurately describing the control and dynamics of the host and intruder aircraft is imperative to predicting collisions in the future. In this section, a brief overview is given of the model used in this thesis as well as a discussion on the implementation of a turbulence module, cross-track controller and altitude controller.

4.3.1 Overview

A Simulink model representative of an aircraft similar to the Airbus A330 class is available to this thesis and is therefore the model used in all test scenarios. The basic model framework can be seen in figure 4.1.

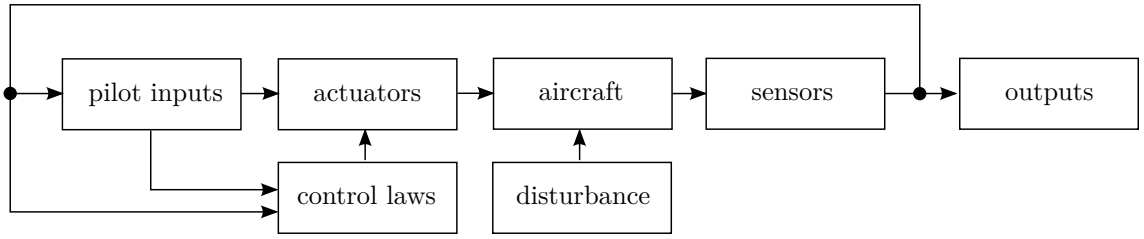


Figure 4.1: Non-Linear Closed-Loop Aircraft Model

The control laws of the aircraft model govern the movement of the aircraft's actuators based on the current states of the aircraft and desired inputs from the pilot. Pilot inputs include those of the side stick, pedals, high lift configuration lever, air brakes and throttle lever.

The aircraft module seen in figure 4.1 corresponds to the basic aircraft model of figure 3.9 on page 44. The aircraft's flight mechanics and performance factors are described herein and are used to create a representation of the current aircraft states given the present state of the aircraft's actuators and environment. The aircraft module in simulation is essentially an approximation of how a real aircraft would react to actuator commands and atmospheric conditions in practice.

While the aircraft module provides a true representation of the current aircraft states, a sensor module is used to model the inaccuracies and uncertainty often present in onboard instruments. The current states are therefore augmented through the sensor module to produce a more realistic, rough estimation of the aircraft's current states. These estimated states are then fed back to the pilot and the control module, where decisions are made based on what the aircraft is currently observed to be doing.

The disturbance present in figure 4.1 is a model of atmospheric conditions, which may perturb the aircraft. For instance, strong winds, pressure fluctuations, temperature changes and turbulence are all factors, which may cause unexpected deviations from the desired control status. The modelling of this disturbance is discussed in more detail in the following section.

The output module in the aircraft model simply provides an interface for the retrieval of flight data. This data lays the foundation for the Monte Carlo simulation and probability flow validation performed in this thesis.

4.3.2 Disturbance

In order to replicate the uncertainty present in the states of a real aircraft, the disturbance due to atmospheric conditions must be modelled. The primary goal of disturbance modelling in this thesis is to facilitate Monte Carlo simulation, which requires the generation of random trajectories.

In reality, if a pilot could exercise the exact same control every time a specific route is flown, differences in the aircraft state values will still occur due to the unpredictable influence of the factors such as the environment, fuel consumption and load for instance. If the aircraft model is considered without disturbance in simulation, the flight data retrieved from the output module is identical over multiple iterations provided that the model inputs remain the same for each simulation.

To ensure that the model behaviour is indeed kept uniform, the model is flown under autopilot conditions. The model in figure 4.1 on the preceding page facilitates autopilot control through the use of a sideslip angle, flightpath angle and yaw angle hold mode. This means that the aircraft model can be initialised with a desired roll, pitch and yaw deflection respectively. The aircraft controllers then attempt to keep the aircraft on par with these angles in the presence of a disturbance until a new hold command is issued.

In this thesis, the effects of pressure fluctuations, temperature changes and constant winds are ignored in favour of a simple turbulence model capable of achieving a relevant envelope of uncertainty for Monte Carlo simulation. Turbulence is ideal for this purpose because it consists of random, unpredictable wind gusts, which, when introduced into the system, will cause the current aircraft states to become stochastic. The uncertainty due to turbulence is therefore characterised by a probability distribution.

There are 2 prominent models when it comes to describing continuous wind gusts, namely the Dryden and Von Kármán wind models. Both models describe turbulence using stochastic processes defined by the vertical, lateral, longitudinal and angular components of velocity. These processes are created by passing band-limited white noise through appropriate forming filters. Different power spectral densities are specified by Dryden and Von Kármán for the band-limited white noise. [49; 50]

A Dryden turbulence model following MIL-F-8785C military specifications was made available to this project by Airbus. The model takes the aircraft's current airspeed, altitude at the centre of gravity, Euler angles and ground track as input from the sensor model. Ground track refers the aircraft's heading in the inertial axis system. The only constant supplied to the model is the wingspan of the aircraft, which, in the case of the Airbus A330 class of aircraft, is 60 m.

The last input of the turbulence model is a vector of 4 noise seeds for the band-limited white noise of the 4 output components seen in figure 4.2 on the next page. Seeds are integers on the interval $[0, (2^{31} - 1)]$ representing different random number streams in MATLAB. In the Airbus turbulence model, seeds are limited to the interval $[1, 365]$. The purpose of seeds is to allow a user to replicate results in simulations, which use band-limited white noise. For Monte Carlo simulation, using the same noise seeds would produce identical turbulence patterns in every simulation. This is undesirable and the 4 noise seeds are therefore randomised by sampling a uniform distribution over the interval $[1, 365]$.

The model outputs turbulence as wind velocity in the inertial axes, denoted by W_{T_N}, W_{T_E} and W_{T_D} respectively, as well as in the aircraft roll axis, denoted by W_{T_P}. It is assumed that only these 4 components are taken into account because they have the most significant impact on the aircraft behaviour. The wind velocities in the inertial axis system are then converted to the principal axis system before being subtracted from the aircraft's current airspeed in the aircraft module of figure 4.1. Similarly, the angular wind velocity in the aircraft roll axis is subtracted from the aircraft's current roll rate. A simulation of the aircraft model with turbulence can be seen in figure 4.3 on the following page.

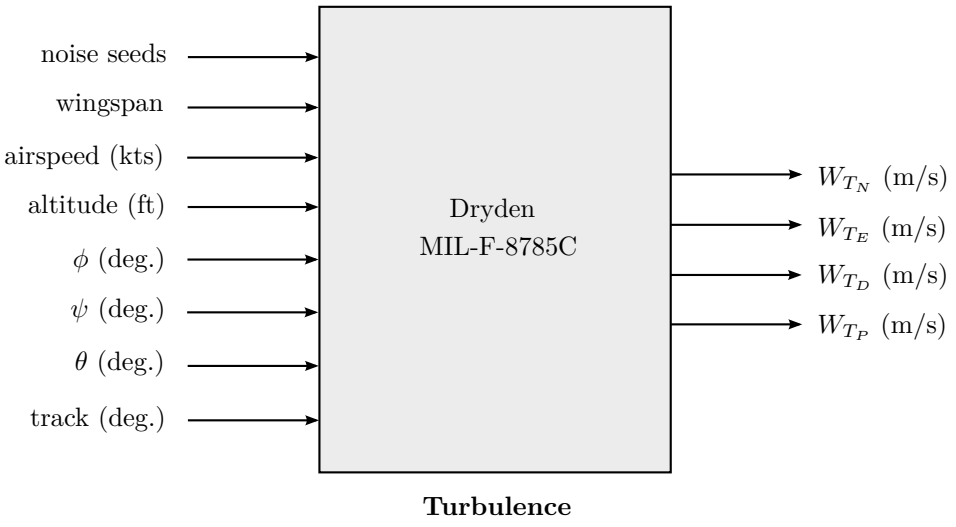
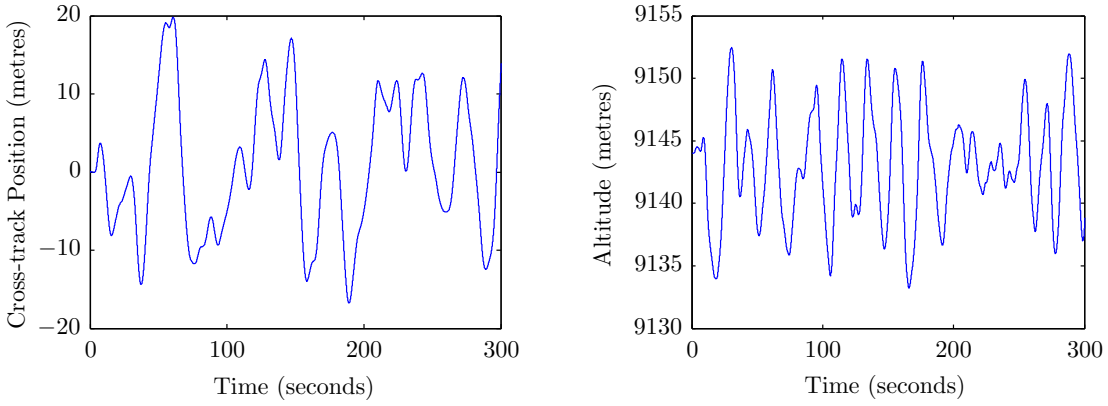


Figure 4.2: Inputs and Outputs of the Dryden Wind Model



(a) Aircraft Cross-track Position over Time

(b) Aircraft Altitude over Time

Figure 4.3: Graphs showing Cross-track Position and Altitude with Disturbance over 5 Minutes

4.3.3 Cross-track Control

When the aircraft model is disturbed under autopilot conditions, the aircraft will deviate from its current trajectory. The autopilot will correct the deviation and return the aircraft to its original hold mode, but it will not return it to its original path. For example, in figure 4.4, the aircraft is commanded to fly straight-and-level. When a wind gust disturbs the aircraft pushing the nose to the right, a deviation from the flight route occurs momentarily before the autopilot compensates for the disturbance returning the aircraft back to straight-and-level flight.

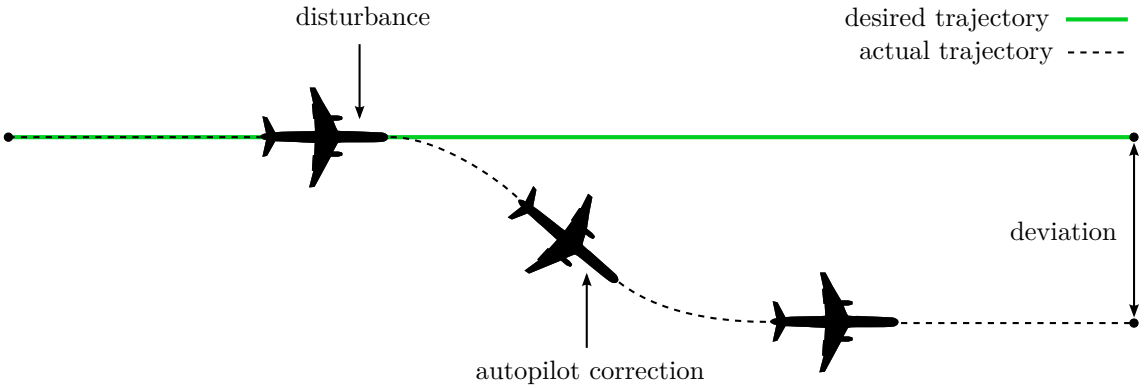
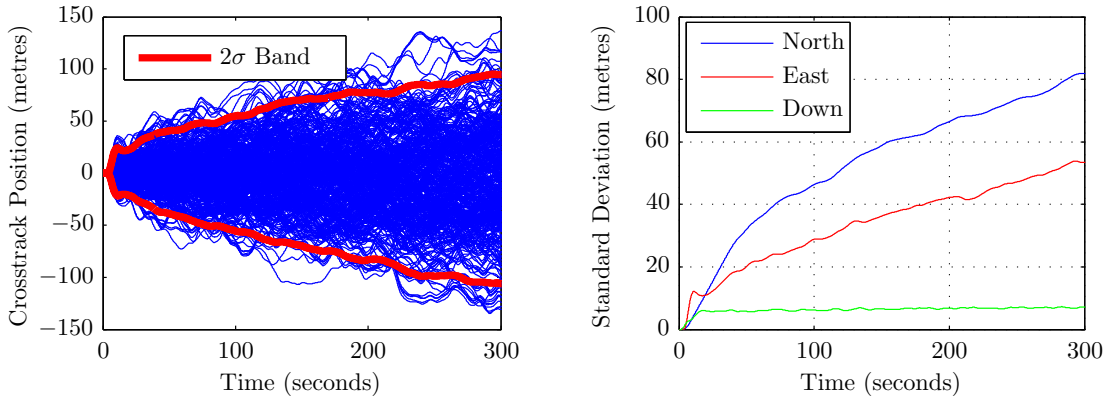


Figure 4.4: Deviation Caused by a Wind Gust

The cross-track deviation shown in figure 4.4 is not corrected by the autopilot. Using the aircraft model, a forward simulation was performed 500 times under autopilot control. The aircraft was commanded to fly straight-and-level for 5 minutes in medium turbulence. The resulting cross-track position over time and the corresponding standard deviation over time is shown in figure 4.5.



(a) Cross-track Position over Time

(b) Standard Deviations over Time

Figure 4.5: Aircraft Cross-track Variability over Time without Cross-track Controller

Increasing cross-track deviation from the desired flight route is observed. In reality, the pilot would either change the autopilot settings or take manual control of the aircraft in a turbulence situation to correct the deviation. To achieve a somewhat similar correction in simulation, a cross-track controller paired with a waypoint scheduler is implemented. It should be noted, however, that the cross-track and altitude controllers presented in this thesis are in no way representative of the ideal controllers required to accurately model the real-world operation of an aircraft. The controllers presented are implemented here solely for the purpose of achieving controllable flight paths in the simulation experiments

to follow.

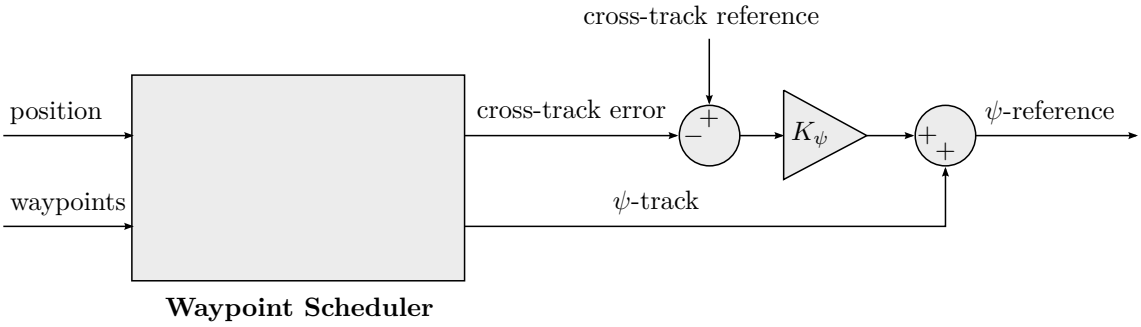


Figure 4.6: Cross-track Controller using Yaw Angle Hold Mode

The cross-track controller makes use of the existing autopilot hold modes to create a consistent envelope of uncertainty around the desired trajectory. This is achieved by producing a heading command proportional to the cross-track error. To obtain the cross-track error, the aircraft position must first be transformed from inertial axis coordinates to a new guidance axis system, depicted in figure 4.7.

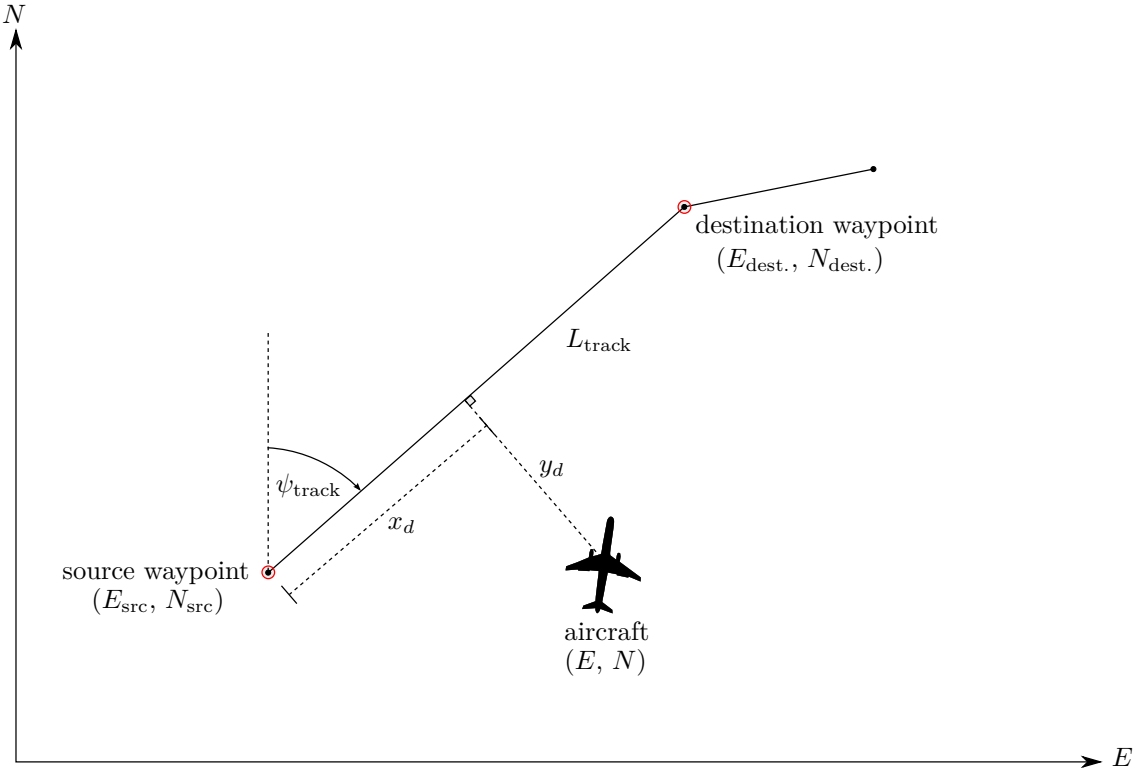


Figure 4.7: Guidance Axis System adapted from [5]

The aircraft's cross-track position, denoted by (E, D), is transformed using

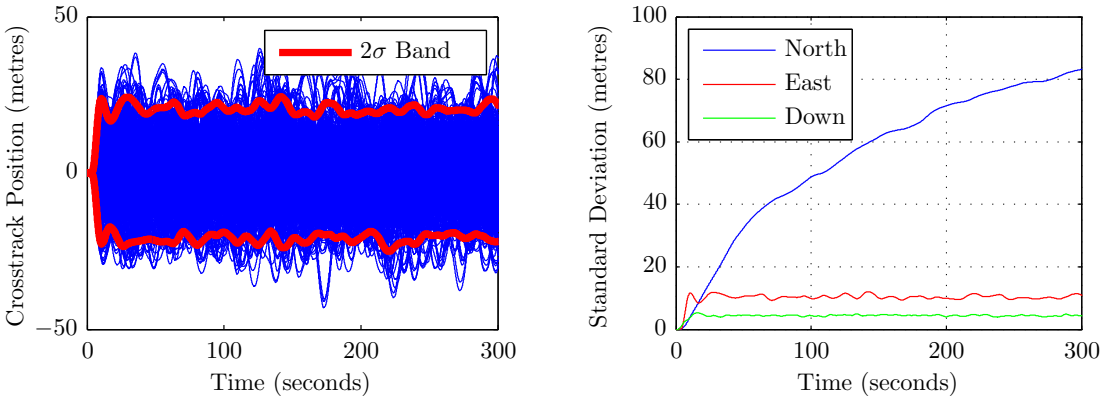
$$\begin{bmatrix} x_d \\ y_d \end{bmatrix} = \begin{bmatrix} \cos(\psi_{\text{track}}) & \sin(\psi_{\text{track}}) \\ -\sin(\psi_{\text{track}}) & \cos(\psi_{\text{track}}) \end{bmatrix} \begin{bmatrix} N - N_{\text{src}} \\ E - E_{\text{src}} \end{bmatrix} \tag{4.3.1}$$

where ψ_{track} is the heading of the flight route and $(E_{\text{src}}, E_{\text{src}})$ is the source (reference) waypoint in inertial axis coordinates. The desired heading is calculated inside the waypoint scheduler using

$$\psi_{\text{track}} = \arctan \frac{E_{\text{dest.}} - E_{\text{src}}}{N_{\text{dest.}} - N_{\text{src}}} \tag{4.3.2}$$

where $(E_{\text{dest}}, N_{\text{dest}})$ is the destination waypoint. The resulting x_d in equation 4.3.1 is the along-track distance from the source waypoint along the flight route and y_d is the cross-track error or distance perpendicular to the flight route.

The cross-track controller is designed to control y_d to a zero reference. The cross-track error is first scaled by a gain value K_ψ before being added to the current heading angle to produce a heading command. The gain value was empirically chosen and adjusted to $K_\psi = 0.03$ where a steady-state cross-track standard deviation was observed in simulation. Finally, the resulting heading command, denoted by ψ-reference, is sent to the yaw angle hold controller to compensate for any cross-track deviatons.



(a) Cross-track Position over Time

(b) Standard Deviations over Time

Figure 4.8: The Effect of Cross-Track Control on Aircraft Position

The waypoint scheduler compares the along-track distance x_d to the distance from the source waypoint to the destination waypoint, denoted by L_{track}, at each time step. L_{track} is calculated using

$$L_{\text{track}} = \sqrt{(N_{\text{dest.}} - N_{\text{src}})^2 + (E_{\text{dest.}} - E_{\text{src}})^2}. \tag{4.3.3}$$

When x_d is larger than L_{track}, the waypoint scheduler makes the source waypoint equal to the destination waypoint and the destination waypoint is incremented. The aircraft model

is therefore now capable of following a desired trajectory within a consistent envelope of uncertainty.

4.3.4 Altitude Control

Altitude control is performed similarly, although independently, to cross-track control. The aim is to create a controller that produces a flightpath angle command proportional to the altitude error. The format of the controller is shown in figure 4.9.

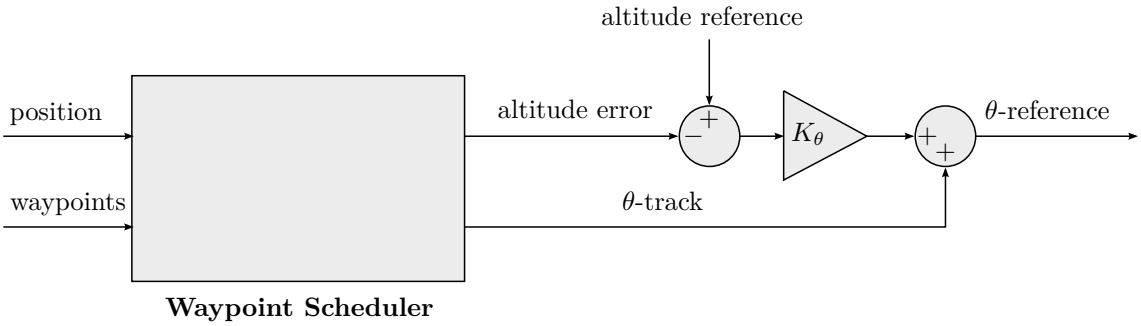


Figure 4.9: Altitude Controller using Flightpath Angle Hold Mode

The guidance axis system showing a view of altitude axis is shown in figure 4.10 on the following page. The desired flight path angle, denoted by θ_{track}, is calculated using

$$\theta_{\text{track}} = \arctan\left(\frac{D_{\text{dest.}} - D_{\text{src}}}{\sqrt{x_d^2 + y_d^2}}\right). \tag{4.3.4}$$

The altitude error z_d can then be calculated using

$$z_d = \tan(\theta_{\text{track}})h_d + g_d \tag{4.3.5}$$

where h_d is the distance in the NE-plane from the source waypoint to the aircraft, calculated by $h_d = \sqrt{x_d^2 + y_d^2}$, and g_d is the difference in altitude from the source waypoint to the aircraft, calculated by $g_d = D - D_{\text{src}}$.

The gain, K_θ, is once again chosen empirically and adjusted until a consistent standard deviation on the aircraft altitude is observed over time.

Before altitude control is employed, the repeated forward simulation of the aircraft model revealed an increasing loss of altitude over time under straight-and-level autopilot conditions (see figure 4.11).The standard deviation of altitude is observed to be fairly consistent, but deviation from the target altitude still increases over time. When turbulence disturbs the aircraft from straight-and-level flight, a shift in lift is experienced, resulting in a loss

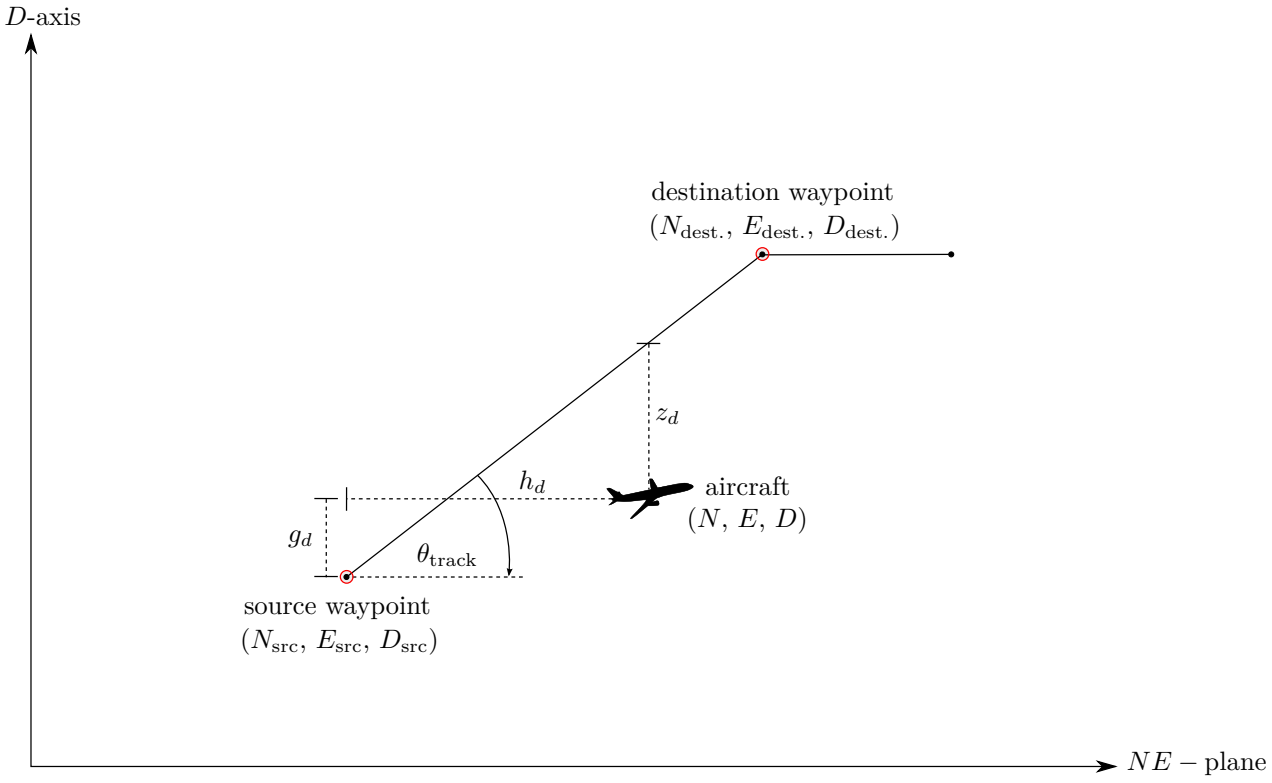
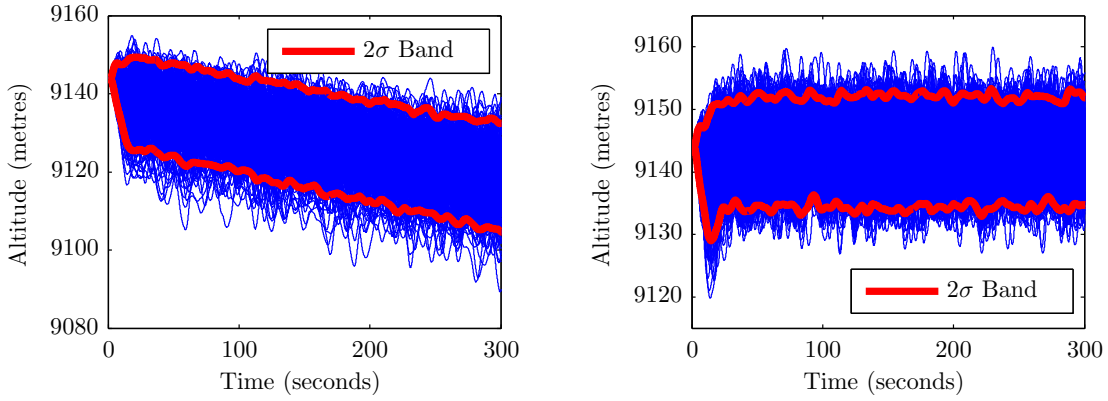


Figure 4.10: Guidance Axis System showing Vertical Plane

of speed and the dominance of gravitational forces. Without an increase in thrust, the aircraft descends over time.

With the application of altitude control, the desired consistent envelope of uncertainty is achieved. Initially, a small loss of altitude is experienced during the transition of the initial trim conditions to a steady-state, but the deviation is quickly corrected in approximately 10 seconds.



(a) Altitude over Time without Altitude Controller

(b) Altitude over Time with Altitude Controller

Figure 4.11: The Effect of Altitude Control on Aircraft Position

4.4 Monte Carlo

The Monte Carlo method is used in this thesis to determine a "ground-truth" result for comparison with the probability flow method of Van Daalen.

4.4.1 Initialisation

To generate a set of random trajectories, the aircraft model states must be initialised with a random sample from the PDF of those states. The initial state PDF is unknown, but it can be determined using a forward-simulation of the model under autopilot conditions with disturbance.

First, the trim values of the model must be found by forward-simulating the open-loop model without turbulence. The open-loop model states are initialised with guessed values informed by the desired inputs of the specific flight scenario, such as initial speed, heading and position. Over time, the states of the open-loop system converge to steady-state values, which are then used as the trim conditions for the closed-loop model.

The closed-loop aircraft model initialised with the trim conditions can now be simulated forward in time with disturbance. Because of the presence of a disturbance, the aircraft states never reach a constant value. The model is therefore simulated N_{MC} times and at the point where the states are observed to be relatively consistent across N_{MC} trajectories, the PDF of those states is calculated. The question is then, how many simulations need to be performed to calculate an accurate representation of the state PDF? It was decided that the value of N_{MC} should be chosen based on the 2σ-rule, which states that 95% of all values must lie within 2 standard deviations of the mean.

It was found that 407 simulations are needed to satisfy this criterion. Equation 4.4.1 can also be used to calculate N_{MC} if an estimate of the standard deviation σ_{MC} is known. A desired error margin E_m is chosen as 1 metre and the z-score associated with a 95% confidence level in a standard normal distribution is denoted by $z_{\alpha/2}$. Running the simulation 50 times revealed an estimate of the standard deviation to be 10.29 metres.

$$N = \left(\frac{z_{\alpha/2}\ \sigma_R}{E_m}\right)^2 = \left(\frac{z(\frac{1-0.95}{2})10.29}{1}\right)^2 = \left(\frac{(1.96)(10.29)}{1}\right)^2 \approx 407 \tag{4.4.1}$$

For each Monte Carlo simulation, the aircraft states are initialised with a random sample from their PDFs. In doing so, it is observed that occasionally a random combination

of initial values that deviate largely from the trim conditions can lead to initial transients in the aircraft response. This is because the aircraft state values cannot be decoupled and the aircraft should in fact be initialised with a sample from the joint PDF of the states.

Calculating the joint state-PDF is a complex process due to the large number of states. This is therefore avoided by only initialising the position states with a random sample from its PDF while the remaining states are initialised at trim. The model is then forward-simulated until the remaining states settle and flight data is discarded up to that point.

Initialising the position state with a random sample allows the aircraft to achieve normally distributed states quicker than if it were initialised at a single point. Less initial flight data therefore has to be discarded from each simulation. Referring to figure 4.8(b) on page 54, it is observed that it takes approximately 10 seconds for the cross-track and altitude standard deviations to reach a steady-state. With a sampling time of 0.01 seconds, this means that the first 1000 samples of every simulation must be discarded.

4.4.2 Calculating Mesh Intersections

For every randomly generated trajectory, it is necessary to determine if an intersection occurs with each meshed conflict region. Each random trajectory is therefore described as a polyline, or series of arbitrary points making up a line. Each polyline is then checked individually for intersection with each triangle face in each mesh.

There exists no simple solution for the intersection between a polyline and a mesh, so the polyline is divided into a series of very small straight-line segments. Each triangular face of the conflict region mesh is then treated as an infinite plane [51]. The intersection between a straight-line and a plane is identified when the parametric line in equation 4.4.2 is evaluated at s_I.

$$\mathbf{p}(s_I) = \mathbf{p_0} + s_I(\mathbf{p_1} - \mathbf{p_0}) \tag{4.4.2}$$

$\mathbf{p_0}$ and $\mathbf{p_1}$ are the start and end points of a line segment respectively. The value of s_I is calculated as follows:

$$s_I = \frac{\mathbf{n_f} \cdot (\mathbf{v_0} - \mathbf{p_0})}{\mathbf{n_f} \cdot (\mathbf{p_1} - \mathbf{p_0})} \tag{4.4.3}$$

The normal vector $\mathbf{n_f}$ is calculated from equation 3.4.2 on page 39 and $\mathbf{v_0}$ to $\mathbf{v_2}$ represent the vertices of a triangular face. When the denominator of equation 4.4.3 is 0, the segment lies parallel to the plane and therefore no intersection occurs. If the denominator is non-zero and real, and $0 \leq s_I \leq 1$, then the segment intersects the plane. Once an intersection has been identified, it must be determined whether or not the intersection point lies within

the triangular face, and not just elsewhere on the plane [51]. The parametric plane equation is given by

$$\mathbf{v}(a,b) = \mathbf{v_0} + a(\mathbf{v_1} - \mathbf{v_0}) + b(\mathbf{v_2} - \mathbf{v_0}) \tag{4.4.4}$$

The point $\mathbf{p}(a,b)$ is inside the triangle only if $a \geq 0$, $b \geq 0$ and $a + b \leq 1$. Equation 4.4.5 is used to calculate a_I and equation 4.4.6 is used to calculate b_I where $\mathbf{u} = \mathbf{v_1} - \mathbf{v_0}$, $\mathbf{q} = \mathbf{v_2} - \mathbf{v_0}$ and $\mathbf{w} = \mathbf{p}(s_I) - \mathbf{v_0}$.

$$a_I = \frac{(\mathbf{u}\cdot\mathbf{q})(\mathbf{w}\cdot\mathbf{q}) - (\mathbf{q}\cdot\mathbf{q})(\mathbf{w}\cdot\mathbf{u})}{(\mathbf{u}\cdot\mathbf{q})^2 - (\mathbf{u}\cdot\mathbf{u})(\mathbf{q}\cdot\mathbf{q})} \tag{4.4.5}$$

$$b_I = \frac{(\mathbf{u}\cdot\mathbf{q})(\mathbf{w}\cdot\mathbf{u}) - (\mathbf{u}\cdot\mathbf{u})(\mathbf{w}\cdot\mathbf{q})}{(\mathbf{u}\cdot\mathbf{q})^2 - (\mathbf{u}\cdot\mathbf{u})(\mathbf{q}\cdot\mathbf{q})} \tag{4.4.6}$$

Pseudocode for the mesh intersection checking process is given below.

Algorithm 1 Mesh Intersection Algorithm

```
1:  total conflict = 0
2:  for each mesh face do
3:      den = denominator of equation 4.4.3
4:      if den != 0 and den is real then si = numerator of equation 4.4.3 / den
5:          if si >= 0 and si <= 1 then
6:              a = equation 4.4.5
7:              if a >= 0 and a <= 1 then
8:                  b = equation 4.4.6
9:                  if b >= 0 and (a + b <= 1) then
10:                     total conflict = total conflict + 1
11:                     detected = true
12:                 end if
13:             end if
14:         end if
15:     end if
16:     if detected == true then
17:         break
18:     end if
19: end for
```

4.4.3 Memory Management

The position and velocity for each randomly generated Monte Carlo trajectory needs to be stored in memory and later used to calculate mean and covariance along the path. The sampling period used in the aircraft model is $T_s = 0.01$. The simulation time is dependent on the scenario, but $T_f = 60$ seconds can be considered a reasonable look-ahead time. The number of samples along the path is therefore $N_S = \frac{T_f}{T_s} = 6000$.

If, for example, 12,000 Monte Carlo simulations are performed in a given scenario, the amount of samples that require storage is 12,000×6000 = 72 million samples, and that is

Table 4.1: Memory Usage Requirements for 12,000 Monte Carlo Simulations with $T_s = 0.01$ and $T_f = 60$

Variable	**Dimensions**	**Instances**	**Memory Usage***
Position	6000 × 3	12,000	6000×3×12,000×8 ≈ 1.61 GB
Velocity	6000 × 3	12,000	6000×3×12,000×8 ≈ 1.61 GB
Mean Position	6000 × 3	1	6000×3×1×8 ≈ 141 KB
Mean Velocity	6000 × 3	1	6000×3×1×8 ≈ 141 KB
Position Covariance	18,000 × 3	1	18,000×3×1×8 ≈ 422 KB
Velocity Covariance	18,000 × 3	1	18,000×3×1×8 ≈ 422 KB

* All variables are stored in double-precision format i.e. 8 bytes per value.

only for a one-dimensional variable. The basic memory requirements for a single scenario are set out in table 4.1.

In MATLAB, the `hold` function was initially used to accumulate data in memory after each simulation. By default, MATLAB stores floating-point numbers in double-precision format, meaning that each number is designated 64-bits (8 bytes) of memory. According to table 4.1, approximately 3.23 GB of memory is required in total.

The computer used for this thesis uses the Windows 7 32-bit operating system with 4 GB of available RAM. The version of MATLAB (R2008a) used allows for a maximum array size of 1281 Megabytes (MB) with a total space of 1566 MB for all arrays. There is therefore insufficient RAM for even the most basic scenario containing only a single aircraft.

To work around this problem, the software was adapted in such a way that every Monte Carlo simulation is stored in a delimited text file – one for velocity and one for position. When the mean and covariance is calculated, simulations are loaded one at a time and subsequently cleared from memory. Unfortunately, this slowed down execution considerably. The software was then once again adapted to use binary files instead of text files, allowing data to be stored in a specified precision format without delimiters and white space characters that cause a bottleneck in the read-write process.

4.5 Probability Flow

Following the introductory discussion in section 2.6.5 on page 27, this section presents the mathematical definition and software implementation of the algorithm.

4.5.1 Definition of a Tight Upper Bound

Recall from section 2.6.5 on page 27 that probability flow is the instantaneous rate of increase in the probability of conflict experienced when a vehicle state PDF is propagated through a conflict region surface boundary. Jones describes how a vehicle's state PDF distorts during the risk accumulation process through the conflict region (see figure 2.6), increasing the complexity of the problem.

The state PDF is therefore assumed to be unchanged with each encounter with the conflict region. This results in the repeated accumulation of risk and an upper bound to the overall probability of conflict. This is achieved by not only considering the set of trajectories at time t, which have not experienced conflict previously, but rather the set of all trajectories in the sample space whether they have earlier experienced conflict or not.

Van Daalen points out that the upper bound will be tight at or below typical threshold values. This is because the size of the upper bound is proportional to the amount of accumulated risk. Particularly, in civil aviation, conflict probability thresholds are extremely low, meaning that for a small predicted risks of conflict, the upper bound will be small or negligible. Kuchar and Yang speculate that a worst-case accuracy of 5% is necessary to differentiate between conflict and no conflict [2].

The definition of the upper bound to the flow of probability in a 3D sample space Ω is defined by Van Daalen as

$$\frac{\mathrm{d}P_C^{UB}(t)}{\mathrm{d}t} = -\iint_{S(C_t)} f_{\mathbf{R}(t)}(\mathbf{p}) \int_{-\infty}^{w_n(\mathbf{p},t)} f_{V_n(t)}\left(v_n | D_t^{\mathbf{p}}\right) v_n \,\mathrm{d}v_n \,\mathrm{d}S. \tag{4.5.1}$$

The full derivation can be found in [13]. Understanding the role of each term in the expression is key to understanding the equation as a whole. Stochastic variables denoted by $\mathbf{R}(t,\omega)$ and $\mathbf{V}(t,\omega)$ describe the vehicle position and velocity states respectively over the interval $[t_0, t]$ for $\omega \in \Omega$. The PDF of $\mathbf{R}(t,\omega)$ given that no conflict condition exists at position variable $\mathbf{p}$ is therefore $f_{\mathbf{R}(t)}(\mathbf{p})$.

The unit normal vector pointing away from the conflict region surface S is denoted by $\mathbf{n}$. The normal component of $\mathbf{V}(t,\omega)$ can then be calculated as $V_n(t,\omega) \triangleq \mathbf{n} \cdot \mathbf{V}(t,\omega)$.

$D_t^{\mathbf{p}}$ is the set of outcomes over $[t_0,t]$ for which $\mathbf{R}(t,\omega)$ is at position $\mathbf{p}$. The PDF of $V_n(t,\omega)$ at velocity variable v_n given $D_t^{\mathbf{p}}$ is therefore $f_{V_n(t)}(v_n|D_t^{\mathbf{p}})$.

The velocity of point $\mathbf{p}$ over $[t_0,t]$ is denoted by $\mathbf{w}(\mathbf{p},t)$. The upper limit of the inner velocity integral in equation 4.5.1 on the previous page is then the normal velocity component of point $\mathbf{p}$ calculated as $w_n(\mathbf{p},t) \triangleq \mathbf{n}(\mathbf{p}) \cdot \mathbf{w}(\mathbf{p},t)$.

Now that an understanding of the different terms in equation 4.5.1 on the preceding page has been established, the method of solving this equation can be discussed. The first step is to assume that the probability distributions of the aircraft position and velocity states are jointly Gaussian.

$$\begin{bmatrix} \mathbf{V}(t,\omega) \\ \mathbf{R}(t,\omega) \end{bmatrix} \sim \left(\begin{bmatrix} \overline{\mathbf{V}}(t) \\ \overline{\mathbf{R}}(t) \end{bmatrix}, \begin{bmatrix} \mathbf{C_V}(t,t) & \mathbf{C_{VR}}(t,t) \\ \mathbf{C_{VR}^T}(t,t) & \mathbf{C_R}(t,t) \end{bmatrix} \right) \tag{4.5.2}$$

Van Daalen then shows that the inner velocity integral of equation 4.5.1 on the previous page can be calculated as

$$-\frac{\sigma_V(t)}{\sqrt{2\pi}} \exp\left(-\frac{\left(\omega_n(\mathbf{p},t) - \overline{V}_n(t)\right)^2}{2\sigma_V^2(t)} \right) + \frac{1}{2}\overline{V}_n(t)\ \mathrm{erfc}\left(\frac{\overline{V}_n(t) - \omega_n(\mathbf{p},t)}{\sqrt{2}\sigma_V(t)} \right), \tag{4.5.3}$$

where the mean normal velocity is

$$\overline{V}_n(t) = \mathbf{n}^T \left(\overline{\mathbf{V}}(t) + \mathbf{C_{VR}}(t,t)\mathbf{C_R}^{-1}(t,t) \left(\mathbf{p} - \overline{\mathbf{R}}(t) \right) \right) \tag{4.5.4}$$

and the velocity variance is

$$\sigma_V^2(t) = \mathbf{n}^T \left(\mathbf{C_V}(t,t) - \mathbf{C_{VR}}(t,t)\mathbf{C_R}^{-1}(t,t)\mathbf{C_{VR}}^T(t,t) \right) \mathbf{n} \tag{4.5.5}$$

at point $\mathbf{p}$ on the interval $[t_0,t]$. The probability of conflict can then be calculated from equation 4.5.1 on the preceding page as

$$P_C^{UB}(t_f) = \int_{t_0}^{t_f} \frac{\mathrm{d}P_C^{UB}(\tau)}{\mathrm{d}\tau} \mathrm{d}\tau. \tag{4.5.6}$$

4.5.2 Summary of Assumptions

The method described by Van Daalen conforms to the following assumptions:

- $P_C(\tau)$ is absolutely continuous on the interval $[t_0, t_f]$.
- The instantaneous probability of conflict at t_0 is negligible. This is important to note because the proposed system must be initialised in a conflict-free situation.
- The probability distributions of the aircraft position and velocity states is jointly Gaussian.
- The aircraft state PDF is unchanged with each encounter with the conflict region.

4.5.3 Adaptive Integration

To efficiently calculate the surface integral in equation 4.5.1 and the time integral in equation 4.5.6, Van Daalen proposes the use of adaptive numerical integration. Adaptive integration, also known as adaptive quadrature, is a method of estimating the value of a definite integral by partitioning the integration domain and then recursively refining only those partitions where the estimate of the integration error is above a certain threshold [13]. An integration rule is used to estimate the value of the integral over each partition. The pseudocode given in algorithm 2 demonstrates how adaptive integration is performed using recursion.

Algorithm 2 Recursive Adaptive Integration Algorithm

```
1:  function adaptiveIntegration(domain, domain_integral)
2:      partition domain into smaller sections
3:      for each section do
4:          section_integral = evaluate integration rule
5:          section_error = calculate error estimate
6:          if section_error > error threshold then
7:              adaptiveIntegration(section, domain_integral)
8:          else
9:              domain_integral = domain_integral + section_integral
10:         end if
11:     end for
12:     return domain_integral
13: end function
```

Both in practice and in simulation, the flight data collected from the aircraft model is uniformly discrete. A family of numerical integration rules called Newton-Cotes formulas is therefore most appropriate for adaptive quadrature. Contrary to Gaussian-based quadrature rules, Newton-Cotes formulas facilitate the estimation of integrals sampled at equally-spaced intervals. [52]

From the Newton-Cotes family, Van Daalen suggests the use of the midpoint rule for the surface integration in equation 4.5.1 and Simpson's rule for the time integration in equation 4.5.6.

The surface being integrated is that of the conflict region described by a polygon mesh of triangles. The integral over a triangle surface in 3D-space is approximated as the product between the area of the triangle and the value of the integrand at the triangle's centroid. Referring to figure 4.12, the midpoint rule is then

$$Q^M_{(\mathbf{p_0},\mathbf{p_1},\mathbf{p_2})} f = A(\mathbf{p_0},\mathbf{p_1},\mathbf{p_2}) f(\mathbf{c_0}), \tag{4.5.7}$$

where $A(\mathbf{p_0},\mathbf{p_1},\mathbf{p_2})$ is the area of the triangle with vertices $\mathbf{p_0}$, $\mathbf{p_1}$ and $\mathbf{p_2}$ and $\mathbf{c_0}$ is the centroid of the triangle. The centroid is calculated using $\mathbf{c_0} = \frac{1}{3}(\mathbf{p_0} + \mathbf{p_1} + \mathbf{p_2})$. The area is calculated using $A(\mathbf{p_0},\mathbf{p_1},\mathbf{p_2}) = \frac{1}{2}|(\mathbf{p_0} - \mathbf{p_1}) \times (\mathbf{p_0} - \mathbf{p_2})|$.

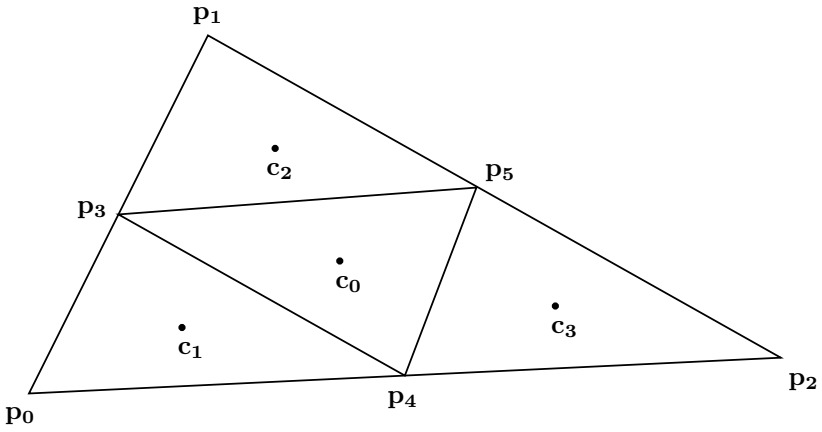


Figure 4.12: Diagram illustrating the Midpoint Rule

To determine an estimate of the error required in the adaptive integration process, the triangle is divided into 4 congruent triangles as shown in figure 4.12. The additional vertices are now defined at $\mathbf{p_3} = \frac{1}{2}(\mathbf{p_0} + \mathbf{p_1})$, $\mathbf{p_4} = \frac{1}{2}(\mathbf{p_0} + \mathbf{p_2})$ and $\mathbf{p_5} = \frac{1}{2}(\mathbf{p_1} + \mathbf{p_2})$. The integration rule is then applied to each of the 4 smaller triangles and the resulting values are summed. The error estimate is then the difference between the integration estimate over the large triangle and the sum of the estimates over the smaller triangles.

$$E^M_{(\mathbf{p_0},\mathbf{p_1},\mathbf{p_2})} f = \left| Q^M_{(\mathbf{p_0},\mathbf{p_1},\mathbf{p_2})} f - \left(Q^M_{(\mathbf{p_0},\mathbf{p_3},\mathbf{p_4})} f + Q^M_{(\mathbf{p_1},\mathbf{p_3},\mathbf{p_5})} + Q^M_{(\mathbf{p_2},\mathbf{p_4},\mathbf{p_5})} + Q^M_{(\mathbf{p_3},\mathbf{p_4},\mathbf{p_5})} \right) \right| \tag{4.5.8}$$

Because the triangle with vertices $\mathbf{p_3}$, $\mathbf{p_4}$ and $\mathbf{p_5}$ has the same centroid as the large triangle, it can be said that $Q^M_{(\mathbf{p_3},\mathbf{p_4},\mathbf{p_5})} f = \frac{1}{4} Q^M_{(\mathbf{p_0},\mathbf{p_1},\mathbf{p_2})} f$. Equation 4.5.8 can therefore be written as

$$E^M_{(\mathbf{p_0},\mathbf{p_1},\mathbf{p_2})} f = \left| \frac{3}{4} Q^M_{(\mathbf{p_0},\mathbf{p_1},\mathbf{p_2})} f - Q^M_{(\mathbf{p_0},\mathbf{p_3},\mathbf{p_4})} f - Q^M_{(\mathbf{p_1},\mathbf{p_3},\mathbf{p_5})} - Q^M_{(\mathbf{p_2},\mathbf{p_4},\mathbf{p_5})} \right|. \tag{4.5.9}$$

If $E^M_{(\mathbf{p_0},\mathbf{p_1},\mathbf{p_2})}f$ is above a specified error threshold, each of the smaller triangles can then be divided into 4 even smaller congruent triangles. The integration estimates for each of those triangles is then calculated to obtain a more accurate estimate of the original integral as well as a lower error estimate. The process can be repeated until the error estimate shrinks to below the threshold. A minimum triangle size can also be specified to terminate the recursion.

A discussion now follows for the calculation of the time integral in equation 4.5.6 using Simpson's rule. Simpson's rule is defined for an integral with integrand $f(\,\cdot\,)$ on the interval $[a,b]$ as

$$Q^S_{[a,b]}f = \frac{b-a}{6}\left[f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right]. \tag{4.5.10}$$

Simpson's rule approximates the integrand by fitting a quadratic polynomial through 3 equally-spaced points (depicted in figure 4.13). The bounds of the initial interval $[a,b]$ make up two of these points. The third point is the midpoint of the interval, which is calculated using $c = \frac{a+b}{2}$.

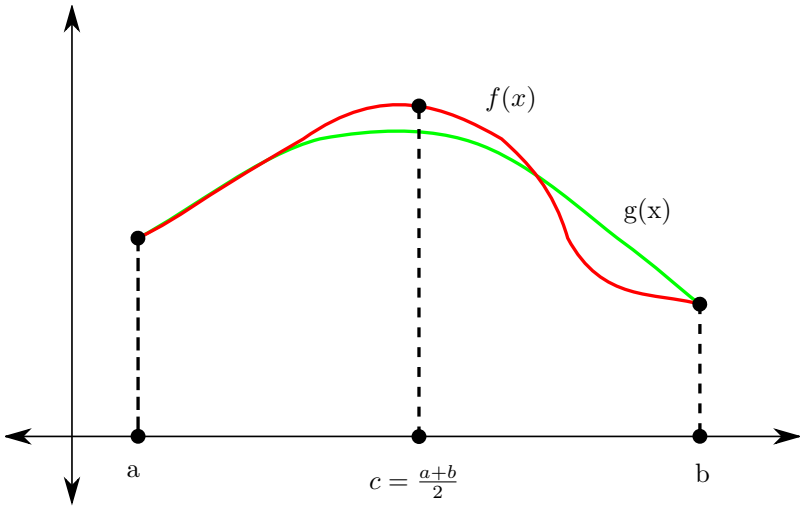


Figure 4.13: Diagram illustrating Simpson's Rule

The error between the true value of the integral and the estimate obtained using Simpson's rule is given by

$$\int_b^a f(x)\mathrm{d}x - Q^S_{[a,b]}f = \frac{1}{90}\left(\frac{b-a}{2}\right)^5 \left|f^{(4)}(\xi)\right| \tag{4.5.11}$$

where $\xi \in [a,b]$. The first step in estimating this error is to evaluate Simpson's rule over both halves of the partitioned domain i.e. $[a,c]$ and $[c,b]$. The sum of these estimates is regarded as the best approximation of the integral over $[a,b]$. The error can then be

calculated using

$$E^S_{[a,b]}f = |Q^S_{[a,b]}f - (Q^S_{[a,c]}f + Q^S_{[c,b]}f)|. \tag{4.5.12}$$

As with the midpoint rule, if $E^S_{[a,b]}f$ is larger than a predefined error threshold, the integration domain is further refined to obtain a more accurate approximation of the integral. The process is repeated until the error estimate is below the threshold. A minimum interval size can also be specified to abort the refinement process.

As proposed by Van Daalen, error thresholds, which are proportional to the time integration interval length or surface integration triangle size are used in this thesis.

4.5.4 Code

Original MATLAB code for a 2D 2-aeroplane scenario and a 3D autonomous underwater vehicle (AUV) scenario was provided by Van Daalen for reference in this thesis. In addition, a C implementation of the probability flow algorithm coupled with a Python interface was included with the 3D example scenario.

The structure of Van Daalen's implementation is set up in such a way that all vehicle and modelling data is pre-defined in MATLAB. The mean and covariance of the AUV position and velocity is found by performing Monte Carlo simulation with a simple linear state-space model using the `lsim` function. The conflict region is assumed to have already undergone Minkowski addition. The MATLAB data is stored in `.mat` files and read by a Python script, which executes a function call to the C program to calculate the probability of conflict.

The provided MATLAB code was studied in order to gain an understanding of the algorithm, but only the C code is reused in this thesis. The latest version of the GNU Compiler Collection (GCC) has removed an option `-mno -cygwin` from the C compiler, but the `distutils` Python package used for creating C extensions has not yet removed the deprecated option. As a result, incompatibility issues arose when attempting the recompile Van Daalen's example code. Finding an older version of GCC is a possibility that was explored, but the `-mno -cygwin` option has also been removed from all previous versions of GCC.

The solution was therefore to remove the Python interface between MATLAB and the probability flow C function. This required the adaptation of the C code and the implementation of a `mex` function to provide the communication link between the two platforms.

Once the code was adapted, it was tested with Van Daalen's 3D scenario initialisation in MATLAB to ensure that the existing results could be replicated.

4.6 Accumulating Risk for Multiple Conflicts

In the event of multiple detected conflicts along a predicted trajectory, the net risk along the path must be calculated. Because it is possible for conflict events to occur simultaneously, it can be said that conflict events are not mutually exclusive. The net risk can therefore not be found by simply adding together the individual probabilities of conflict along the path. The net risk must calculated by finding the probability that at least 1 conflict event occurs. This is in essence the probability of the union of all conflict events.

$$P_{C_T} = P\left(\bigcup_{k=1}^{N_O} C_k\right) \tag{4.6.1}$$

The net risk along the predicted trajectory is denoted by P_{C_T} and N_O is the total number of obstacles including intruder aircraft, terrain, weather or protected airspace. The inclusion-exclusion principle stated in equation 4.6.2 [53] for $N_O \geq 2$ can be used to determine P_{C_T}.

$$P\left(\bigcup_{k=1}^{N_O} C_k\right) = \sum_{m=1}^{N_O}\left((-1)^{m-1} \sum_{\substack{K\subset\{1,\ldots,N_O\}\\|K|=m}} P\left(\bigcap_{k\in K} C_k\right)\right) \tag{4.6.2}$$

The last sum in the equation continues over all subsets K containing exactly m elements. In more general form, it can be written out as

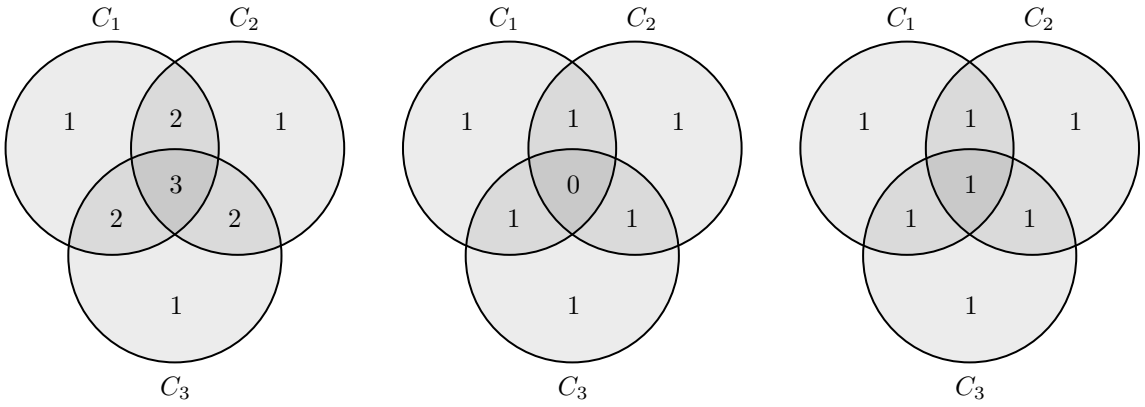
$$\begin{aligned} P\left(\bigcup_{k=1}^{N_O} C_k\right) = &\sum_{k=1}^{N_O} P_{C_k} - \sum_{k<l} P(C_k \cap C_l) + \sum_{k<l<m} P(C_k \cap C_l \cap C_m) - \ldots \\ &+ (-1)^{N_O-1} P\left(\bigcap_{k=1}^{N_O} C_k\right). \end{aligned} \tag{4.6.3}$$

For 2 conflict events, C_1 and C_2, this is simple to compute.

$$P_{C_T} = P(C_1 \cup C_2) = P_{C_1} + P_{C_2} - P(C_1 \cap C_2) \tag{4.6.4}$$

The principle is, however, demonstrated more clearly with 3 events.

$$\begin{aligned} P_{C_T} &= P(C_1 \cup C_2 \cup C_3) \\ &= P_{C_1} + P_{C_2} + P_{C_3} - P(C_1 \cap C_2) - P(C_1 \cap C_3) - P(C_2 \cap C_3) \\ &+ P(C_1 \cap C_2 \cap C_3) \end{aligned} \tag{4.6.5}$$



(a) $|C_1| + |C_2| + |C_3|$

(b) $|C_1| + |C_2| + |C_3| - |C_1 \cap C_2| - |C_1 \cap C_3| - |C_2 \cap C_3|$

(c) $|C_1| + |C_2| + |C_3| - |C_1 \cap C_2| - |C_1 \cap C_3| - |C_2 \cap C_3| + |C_1 \cap C_2 \cap C_3|$

Figure 4.14: Venn Diagrams for Illustration of the Inclusion-Exclusion Principle

The Venn diagrams in figure 4.14 illustrate the union of 3 conflict events at 3 different stages of the inclusion-exclusion process. The numbers indicated in each circle section represent the number of times the associated portion of risk is accumulated.

In this thesis, it is assumed that the occurrence of a conflict does not influence the likelihood of an additional conflict occurrence. Conflict events are therefore said to be independent of one another. The probability of the intersection of conflict events can accordingly be calculated by multiplying the individual conflict probabilities.

$$P\left(\bigcap_{k=1}^{N_O} C_k\right) = \prod_{k=1}^{N_O} P_{C_k} \tag{4.6.6}$$

From equation 4.6.3, it is observed that calculating the net probability of conflict becomes a cumbersome process for many conflict events. Algorithm 3 demonstrates how the process is streamlined using recursion.

Algorithm 3 Recursive Inclusion-Exclusion Algorithm

```
function multipleProbUnion(probvec)
    if length(probvec) == 1 then
        return probvec
    else
        a = probvec(1)
        b = multipleProbUnion(probvec(2:length(probvec)))
        return a + b − (a ∗ b)
    end if
end function
```

When the principle of inclusion-exclusion is used with upper bounded probabilities calculated using probability flow, an upper bound is also expected on the net probability of conflict. The net risk will, however, have no upper bound if an upper bound is not present on any of the individual conflict probabilities along the predicted trajectory.

This concept can be explained by first defining the overestimate on a probability of conflict as

$$\Delta P_{C_k} = P_{C_k}^{UB} - P_{C_k} \tag{4.6.7}$$

where $k \in \{1, 2, \ldots, N_O\}$, $P_{C_k}^{UB}$ is an upper bounded conflict probability and P_{C_k} is the true probability of conflict obtained through Monte Carlo simulation. When the net risk is calculated using upper bounded probabilities, equation 4.6.1 becomes

$$P_{C_T}^{UB} = P^{UB}\left(\bigcup_{k=1}^{N_O} C_k\right). \tag{4.6.8}$$

In order for there to be an upper bound on the net probability of conflict, the probability of the union of all conflict probability overestimates must be larger than 0. From the definition of the union, it can be concluded that there will always be a non-zero overestimate on the net probability of conflict provided that an upper bound is present on at least 1 conflict probability along the path. The overestimate can also never be larger than the sum of all individual conflict probability overestimates.

Chapter 5

Simulations and Results

The Monte Carlo and probability flow methods are implemented in 4 key simulation scenarios. A description of each scenario's implementation, the motivation behind their selection and the results of each simulation are presented in this chapter. The performance of the probability flow method is observed in the civil aviation context and compared to a "ground-truth" result obtained through Monte Carlo simulation.

5.1 Scenario 1: Two-Aeroplane Example

This scenario is an adaptation of a well-known conflict detection example first introduced by Paielli and Erzberger [14], and later also used by Yang et al. [2], Jones [26] and Van Daalen [13]. While these authors present an implementation for the two-dimensional case, this thesis presents the scenario in three-dimensions where both aircraft are flying at the same altitude. The scenario is not repeated here in two-dimensions because probability flow has already been proven by Van Daalen to perform this scenario more efficiently and with the least error in comparison with the other methods [13]. This scenario is, however, implemented in 3D to demonstrate that probability flow is capable of predicting a simple mid-air collision as adequately as TCAS would be able to.

In this flight scenario, the host aircraft is flying straight-and-level at a constant velocity of 240 knots (123.5 m/s) East while a single intruder flies straight-and-level at 240 knots North. The aircraft is simulated under autopilot conditions with a medium turbulence disturbance setting. The aircraft starting positions are staggered, which results in a near-miss collision and a potential violation in safe separation criteria.

A conceptual diagram of the scenario initialisation is shown in figure 5.1 on the next page. The radius of both the host and intruder aircraft conflict regions is chosen to be equivalent to the 60 metre wingspan of an Airbus A330 aircraft. The Monte Carlo simulation of the host and intruder aircraft is performed separately for $N_{MC} = 12{,}000$. This number was chosen to be sufficiently large to ensure a negligible margin of uncertainty on the

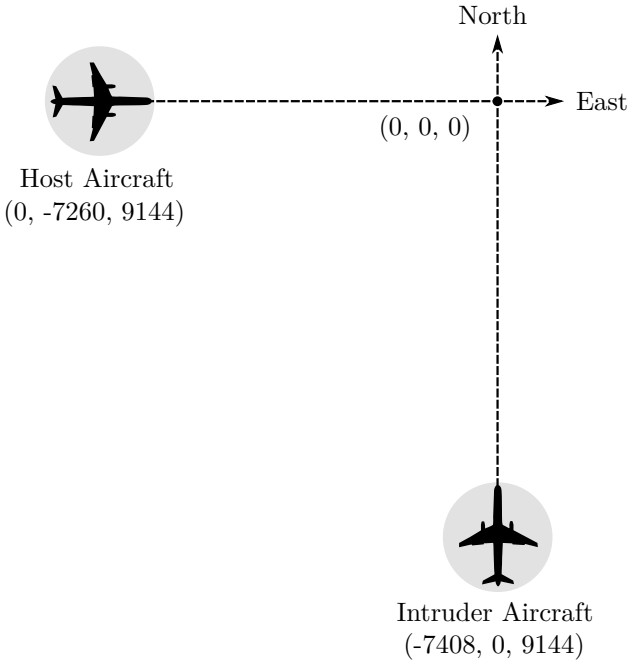
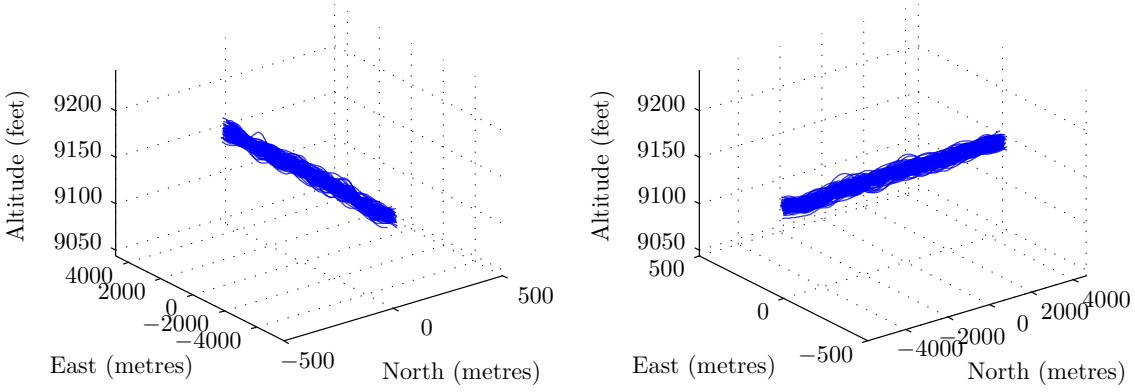


Figure 5.1: Two-Aeroplane Example Conceptual Diagram

"ground-truth" result, but also sufficiently small so as to avoid unnecessarily large data files. Figure 5.2 shows 50 Monte Carlo trajectories (limited for display purposes) for the host and intruder aircraft.



(a) Host Aircraft Trajectories (b) Intruder Aircraft Trajectories

Figure 5.2: Monte Carlo Simulation of Two-Aeroplane Scenario

The relative Monte Carlo trajectories of the intruder to the host are calculated by subtracting each point along the host's path from each point along the intruder's path corresponding in time for each simulation. The resulting trajectories are shown in figure 5.3 on the following page.

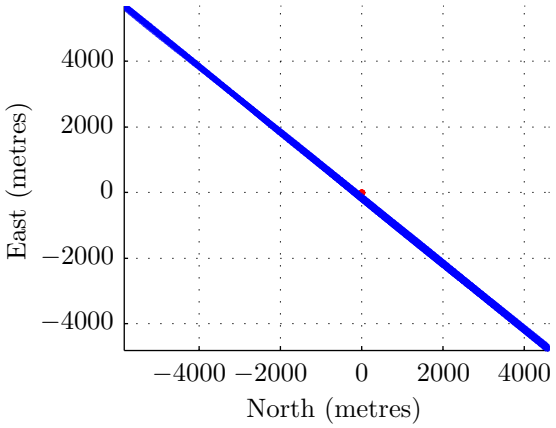


Figure 5.3: Scenario 1: Top-View of Relative Monte Carlo Trajectories

The red dot at the origin represents the combined spherical conflict regions of the host and intruder aircraft. The number of relative trajectory intersections with the sphere divided by the total number of Monte Carlo simulations (12,000) is the probability of conflict P_C.

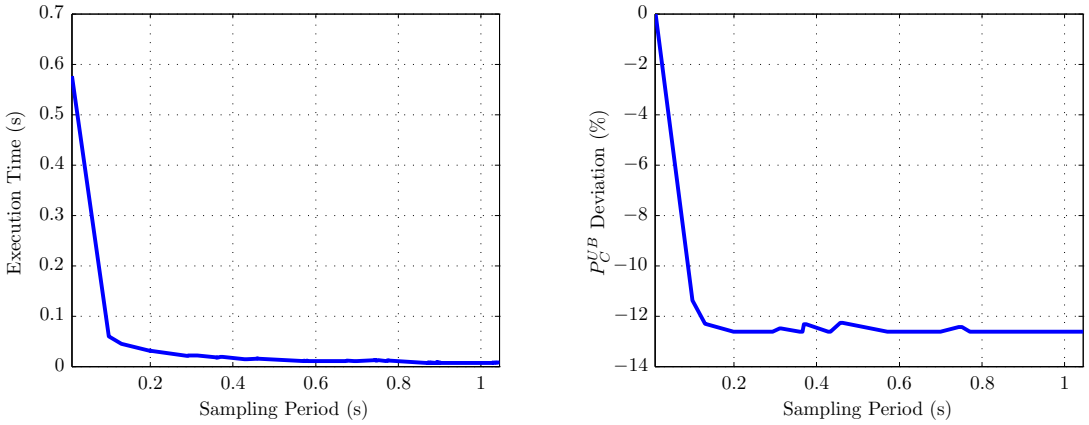
The probability of conflict and associated uncertainty for the two-aeroplane scenario for a look-ahead time of 60 seconds is given in table 5.1. The execution times of both Monte Carlo and probability flow methods are provided, but cannot be realistically compared as they are implemented on two different platforms, MATLAB and C respectively. Execution time would no doubt be improved with code optimisation, parallel processing and a faster computer. It should however be noted that the probability flow algorithm executes in only 0.56 seconds even without any efficiency improvements.

Table 5.1: Two-Aircraft Scenario Simulation Results

Method	P_C (%)	**Error*** (%)	**Uncertainty**† (%)	**Running Time (s)**
Monte Carlo	12.433	-	0.903	94040.85
Probability Flow	12.621	0.188	0.503	0.56

* compared to Monte Carlo simulation
† 3 $\times$ standard deviation for Monte Carlo, total estimated integration error for probability flow

Van Daalen postulates that if the algorithm's execution time is less than 100 ms, then the algorithm can be said to execute in real-time. This scenario therefore clearly does not meet this requirement. This is attributed to the 0.01 second sampling period required by the aircraft model paired with a look-ahead time of 60 seconds, resulting in a vast number of samples. Figure 5.4 on the next page depicts the effect of varying the sampling period on the execution time and accuracy of probability flow.



(a) (b)

Figure 5.4: Scenario 1: Effect of Sampling Period on Execution Time and Probability of Conflict

The execution time does indeed decrease for larger sampling periods, but the loss of accuracy is severe. The P_C^{UB} deviation is defined as the variation of the probability of conflict calculated using probability flow from the most accurate obtained value. In figure 5.4(b), this is the deviation from P_C^{UB} calculated with a 0.01 second sampling period. With sampling periods larger than 0.2 seconds, it can be seen that almost zero conflict is detected. The duration of the conflict between the host and intruder aircraft is therefore extremely small, only 1.2 seconds as seen in figure 5.5, and is undetected between larger sampling periods. This is possible with scenarios in which aircraft are travelling at high speeds, the conflict is very slight and the duration of conflict is minuscule. For this scenario, it is therefore recommended that the sampling period be kept at 0.01 seconds. It may be possible, however, to increase the sampling period to achieve a quicker execution time in scenarios near airports where aircraft travel at lower speeds.

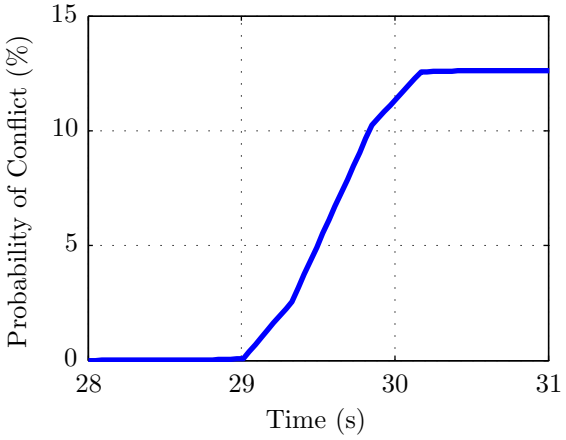
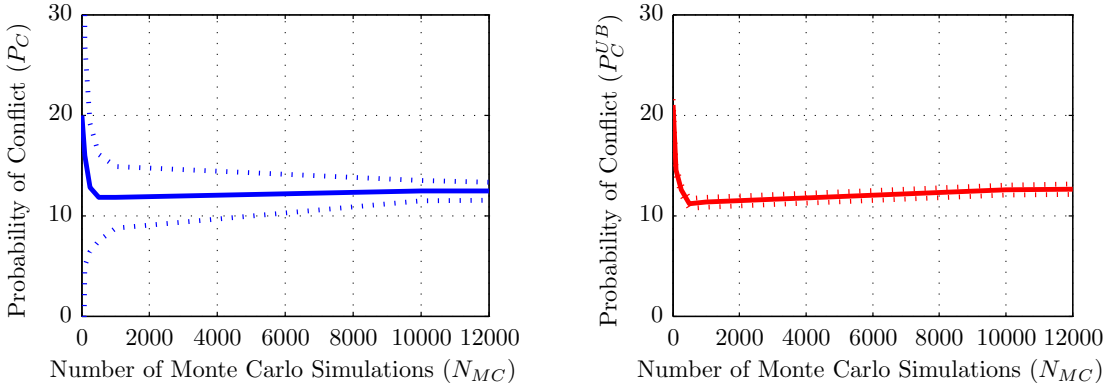


Figure 5.5: Scenario 1: Accumulation of Risk over Time

From table 5.1, an overestimate of 0.188 % is obtained on the probability of conflict. However, there is uncertainty of $\sigma_{MC} = 0.301$ % on the probability of conflict calculated with Monte Carlo simulation as well as an error estimate of $E^M(\text{surface}) + E^S(\text{time}) = 0.503$ % on the probability of conflict calculated with the probability flow method. According to the 3σ-rule, it can therefore be said that 99.73 % of the true probability of conflict lies on the interval $[11.53, 13.336]$ % and the upper bound to the probability of conflict lies on the interval $[12.118, 12.809]$ %. Consequently, the possible error on the true probability of conflict lies on the interval $[0, 1.279]$ %.

In scenarios like the two-aeroplane example where the host aircraft and the conflict region diverge quickly after a conflict occurs, the overestimate on the probability of conflict will be very small or non-existent. This is because repeated accumulation of risk does not occur as it does in scenarios where the host aircraft is exposed to the conflict region for longer periods of time. The overestimate on the probability of conflict in this scenario is consequently not large enough to influence a resolution decision.



(a) P_C versus N_{MC} calculated using Monte Carlo simulation

(b) P_C^{UB} versus N_{MC} calculated using probability flow

Figure 5.6: Scenario 1: Convergence of P_C with increasing values of N_{MC}

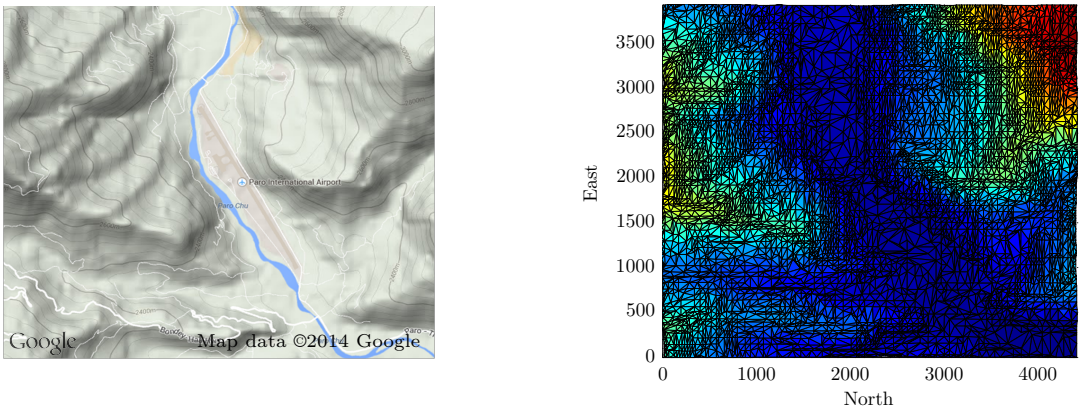
In figure 5.6, the convergence of the probability of conflict for both Monte Carlo and probability flow methods is plotted for increasing values of N_{MC}. From this, it is confirmed that the probability of conflict indeed becomes more accurate and less uncertain as the number of simulations tends to infinity. The upper bound to the probability of conflict, however, has a relatively consistent error bound over N_{MC} simulations because the integration error thresholds are adjusted relative to N_{MC} for each simulation.

Although probability flow does not execute in real-time according to Van Daalen's 100 ms criterion, the method still executes in under 1 second. To the author's knowledge, there is no published measurement of the speed at which TCAS can detect a conflict, only that

surveillance updates are performed once per second. According to Kuchar and Drumm [54], pilots normally take 5 to 15 seconds to react to an RA after initially receiving the warning. It can therefore be concluded that probability flow is a viable method for this type of scenario, allowing at least 59 seconds for a resolution manoeuvre.

5.2 Scenario 2: Terrain Only

This scenario consists of a single host aircraft descending slowly through a valley for 42 seconds. This is a realistic example because the particular valley chosen contains the Paro airport in Bhutan, at the eastern end of the Himalayas. This scenario was chosen to test if probability flow can predict CFIT accidents similarly to EGPWS in a dangerous airport environment. A Google map and corresponding top-view mesh of the terrain is shown in figure 5.7.



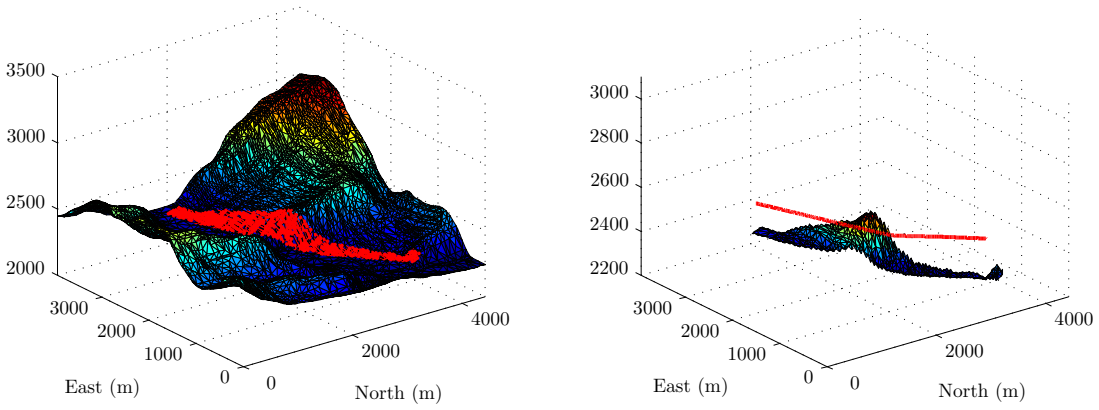
(a) Google Map (b) Top-View Mesh

Figure 5.7: Terrain Maps of Paro, Bhutan Airport

The simulation is initialised with the host aircraft at a position of 4.85 km North and 0 km East at an altitude of 2.48 km in the inertial axis system. The target velocity is specified to start at 154 knots and uniformly decelerate to 150 knots along the path. Once again, Monte Carlo simulation is performed for $N_{MC} = 12{,}000$ under medium turbulence conditions.

To speed up computation time, the mesh faces that lie outside 3 standard deviations of the mean Monte Carlo trajectory are discarded. This margin is chosen to correspond with the 3σ-rule, which states that nearly all values (99.73 %) lie within 3 standard deviations of the mean in a normal distribution. Figure 5.8 on the next page depicts the first step in

the mesh reduction process.



(a) Mesh Vertices within $3\sigma_{MC}$ of Host Trajectory

(b) Reduced Mesh

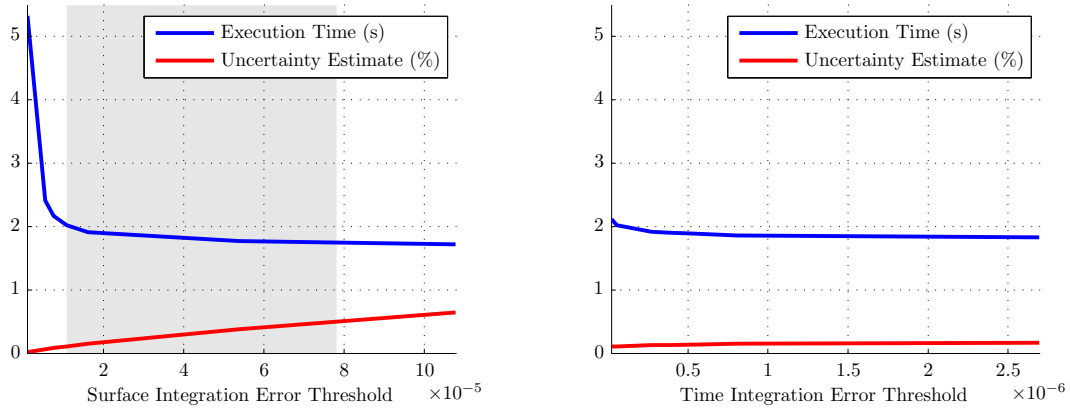
Figure 5.8: Scenario 2: Mesh Reduction to Decrease Computation Time

The second step, is the simplification of the mesh by approximating areas of low detail with larger triangle faces. This is achieved through the use of the `reducepatch` function in MATLAB. The function attempts to reduce the number of mesh faces to a specified percentage of the initial number while still preserving the overall shape of the original object. The `reducepatch` function must, however, be used with caution as reducing the mesh too much will result in a severe loss of accuracy.

With regards to probability flow, the ideal integration error threshold range for this scenario must first be calculated before the optimal mesh reduction factor can be determined. This is because the surface and time integration error thresholds are chosen to be inversely proportional to the number of triangles in the terrain mesh. Figure 5.9 on the following page shows the effect of varying the surface integration error threshold on the interval $[1.077 \times 10^{-7}, 0.1077]$ and the time integration error threshold on the interval $[5.3848 \times 10^{-10}, 5.3848 \times 10^{-4}]$.

The time integration error threshold is kept constant in figure 5.9(a) while the surface integration error threshold is kept constant in 5.9(b). The corresponding integration error estimates are therefore approximately constant meaning that any disparity in the total estimate of uncertainty is primarily due to the variable threshold in question.

From figure 5.9(a) on the next page, it is decided that an execution time below 2 seconds is desirable while the total estimate of uncertainty on the upper bound to the probability of conflict should be kept below 0.5 %. The ideal surface integration error threshold



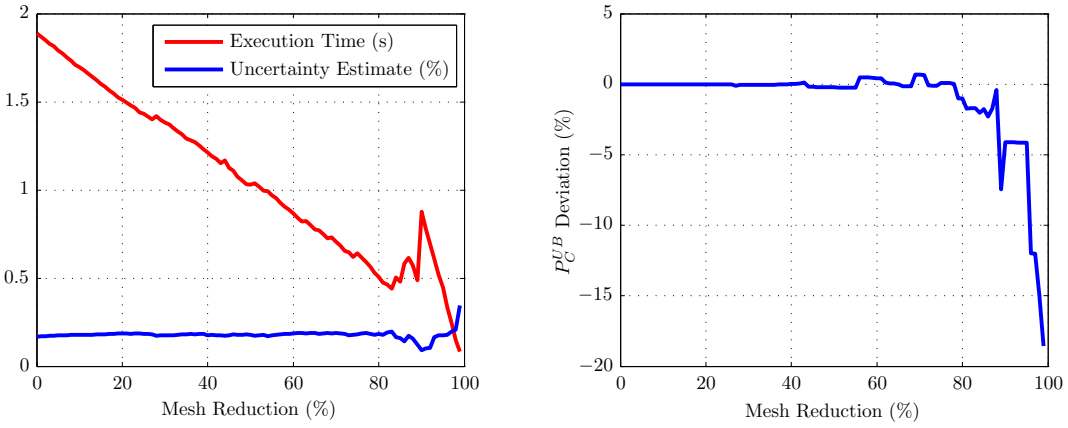
(a) (b)

Figure 5.9: Scenario 2: Effect of Integration Error Thresholds on Execution Time and Uncertainty

range is therefore on the interval $[1.77\times10^{-5}, 7.8131\times10^{-5}]$, shown in grey on figure 5.9(a).

Looking at figure 5.9(b), it is observed that varying the time integration error threshold has little effect on both execution time and total uncertainty. This is likely due to the tight sampling period of 0.01 seconds in the aircraft model. The threshold is therefore chosen to be 5.3848×10^{-6} where the execution time appears to flatten out below 2 seconds.

The effect of varying mesh reduction factors can now be observed in figures 5.10 and 5.11 on the next page. Due to the inverse proportionality of the number of triangles to the integration error thresholds, the surface integration error threshold is chosen as 1.77×10^{-5} on the lower side of the ideal spectrum, leaving room for the threshold to increase as the number of triangles is decreased.



(a) (b)

Figure 5.10: Scenario 2: Effect of Mesh Reduction on Execution Time, Uncertainty and P_C^{UB}

In figure 5.10(a) on the preceding page, an expected decrease in computation time is observed as well as a slight increase in uncertainty for large percentages of mesh reduction. In figure 5.10(b), a large deviation in the upper bound to the probability of conflict can be seen for a mesh reduction between 80 % and 100 %. This is important to note because the oversimplification of a mesh could underestimate the surface integration error and thus underestimate the overall probability of conflict.

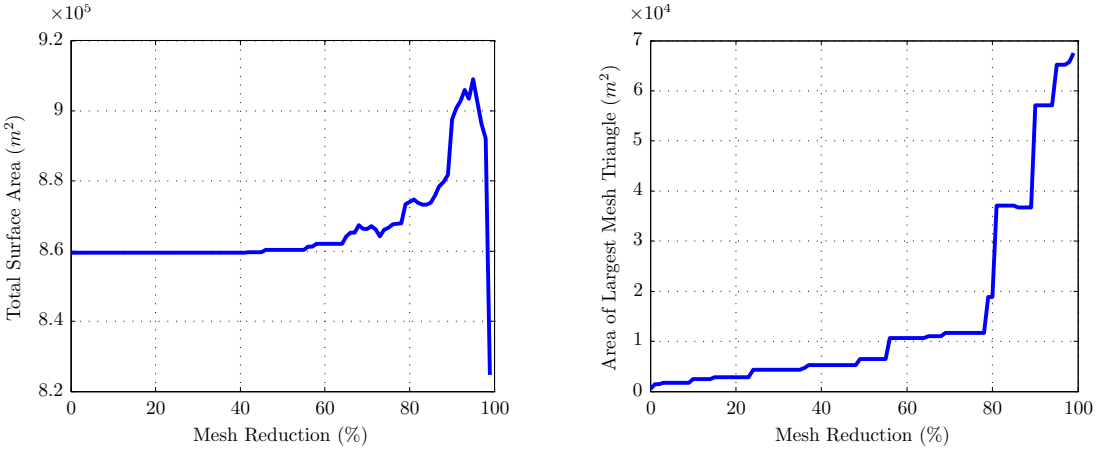


Figure 5.11: Scenario 2: Effect of Mesh Reduction on Surface Area and Maximum Triangle Size

Van Daalen recommends that the maximum size of mesh triangles therefore be constrained by the smallest singular value of the covariance matrix of the aircraft position PDF [13]. This ensures that significant amounts of probability flow are not missed due to an excessive distance between the aircraft position distribution on the mesh surface and any triangle centroid. For this scenario, a mesh reduction of at most 78 % can be performed, resulting in at most an underestimate of 0.2452 % on PC_C^{UB} and a maximum triangle size of 11705 m^2. Probability flow can therefore be optimally executed in 0.39 seconds for this scenario. The results are shown in table 5.2.

Table 5.2: Terrain Only Scenario Simulation Results (first iteration)

Method	P_C (%)	**Error*** (%)	**Uncertainty†** (%)	**Running Time (s)**
Monte Carlo	2.25	-	0.405	57442.44
Probability Flow	4.346	2.096	0.0929	0.39

* compared to Monte Carlo simulation
† 3 × standard deviation for Monte Carlo, total estimated integration error for probability flow

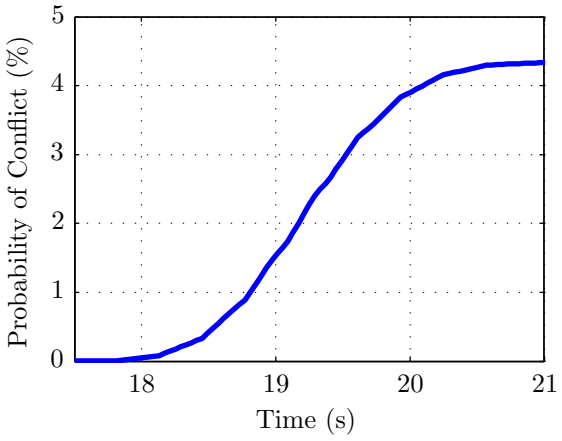


Figure 5.12: Scenario 2: Accumulation of Risk over Time

The overestimate of 2.096 % on a Monte Carlo probability of conflict of 2.25 % is a large error on such a small probability. To obtain a clearer understanding of the problem, the simulation was therefore repeated with the planned trajectory of the host moved closer to the right-hand side of the valley, resulting in a higher risk of conflict. The results can be seen in table 5.3.

Table 5.3: Terrain Only Scenario Simulation Results (second iteration)

Method	P_C **(%)**	**Error*** **(%)**	**Uncertainty†** **(%)**	**Running Time (s)**
Monte Carlo	23.61	-	1.164	47863.59
Probability Flow	42.361	18.751	0.0584	0.56

* compared to Monte Carlo simulation
† 3 × standard deviation for Monte Carlo, total estimated integration error for probability flow

Once again, a large overestimate on the probability of conflict is present. It is suspected that this is because the host aircraft flies parallel to the mountain in this scenario, resulting in long exposure to the conflict region. To understand this effect, the perpendicular and parallel flow of probability through obstacles is compared.

5.2.1 Flow Perpendicular to Obstacles

As already confirmed in the two-aeroplane example, when an aircraft meets an obstacle head-on (perpendicularly), the overestimate on the probability of conflict obtained through the probability flow method will be very small or non-existent. This is because after the aircraft position and velocity PDF has been distorted, it moves further into the conflict region. The intersection surface area between the PDF and the conflict region therefore

remains approximately the same. The concept is demonstrated graphically in figure 5.13.

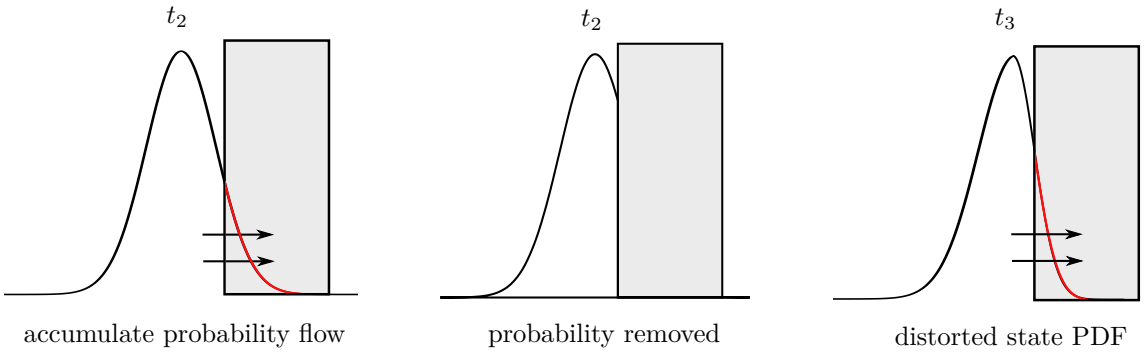


Figure 5.13: Tight Upper Bound for Perpendicular Probability Flow

The movement of the vehicle state PDF towards the conflict region is not shown in the figure, but occurs at t_1. At t_2, the PDF is propagated into the conflict region. The flow of probability is accumulated across the conflict region boundary and removed from the PDF. The PDF is therefore distorted at t_3. The PDF tries to return to Gaussian form by shifting some outcomes into the empty space created by the removal of probability, but the distortion moves the boundary of contact away from the conflict region, decreasing the surface area across which probability is accumulated.

However, at this same instance, the distorted PDF is propagated further into the conflict region. This moves the boundary of contact towards the conflict region again, increasing the accumulation surface area. The resulting surface area is therefore roughly equivalent at t_2 and t_3 or negligibly smaller at t_3, as shown in red. If the rate of probability increase is measured across the same contact area over time, the PDF can then be assumed unchanged with each iteration of propagation.

To test this theory outright, a simple scenario in which the host aircraft flies perpendicularly into the conflict region (a flat wall) at 240 knots is simulated. The wall is chosen to be smaller than 2 standard deviations of the mean Monte Carlo trajectory to ensure that the true probability of conflict is not 100 %, so that any overestimate obtained through probability flow can be observed. The results are shown in table 5.4 on the following page.

From the results, it can be clearly seen that the error of -0.049 % is extremely small. With a standard deviation of 0.388 % on the Monte Carlo probability of conflict, the true probability of conflict lies on the interval $[22.561, 24.889]$ and the probability of conflict calculated using probability flow is on the interval $[23.6196, 23.7324]$. The overestimate

Table 5.4: Perpendicular Conflict Simulation Results

Method	P_C (%)	**Error*** (%)	**Uncertainty**† (%)	**Running Time (s)**
Monte Carlo	23.725	-	1.164	7252.65
Probability Flow	23.676	-0.049	0.0564	2.05

* compared to Monte Carlo simulation
† 3 $\times$ standard deviation for Monte Carlo, total estimated integration error for probability flow

to the probability of conflict therefore lies on the interval $[0, 1.1566]$ %. A maximum overestimate of 1.1566 % is very small and therefore confirms the theory. It should be noted that the execution time of 2.05 seconds is larger here than in the scenario initially presented in this section because no mesh optimisation techniques were performed on the wall.

5.2.2 Flow Parallel to Obstacles

Using figure 5.14, the effect of flying closely parallel to an obstacle on the accumulation of probability flow can be explained. After moving towards the conflict region at t_1, the aircraft state PDF experiences conflict at t_2. At this same instant, probability flow is accumulated and removed from the PDF. Once again, the PDF tries to return itself to the shape of a Gaussian distribution, but the PDF is left distorted nonetheless.

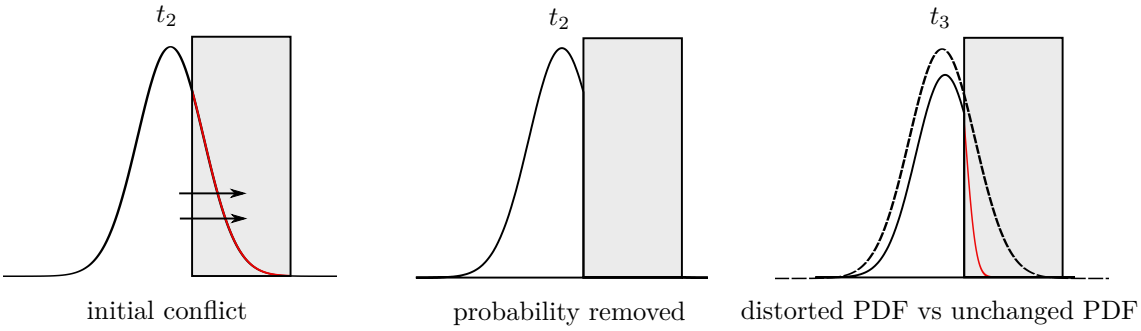


Figure 5.14: Overestimate for Parallel Probability Flow

At t_3, the PDF is not propagated further into the conflict region, nor does it diverge. Flying parallel to the conflict region results in the repeated distortion of the aircraft state PDF over time at the same point of contact. This creates a smaller intersection surface between the aircraft state PDF and the conflict region after each time step of propagation (shown in red). If it is assumed that the aircraft state PDF remains unchanged with

each encounter with the conflict region (shown as dotted), probability flow is accumulated across a greater contact area than it should be, resulting in a very large overestimate on the probability of conflict. The magnitude of the overestimate is dependent on the shape of the terrain and duration of conflict.

A simple scenario to demonstrate this concept was performed for a host aircraft flying straight-and-level at 240 knots parallel to a wall. The results of this scenario are shown in table 5.5.

Table 5.5: Parallel Conflict Simulation Results

Method	P_C (%)	**Error*** (%)	**Uncertainty†** (%)	**Running Time (s)**
Monte Carlo	2.508	-	0.429	7687.5
Probability Flow	8.308	5.8	0.0842	2.17

* compared to Monte Carlo simulation
† 3 × standard deviation for Monte Carlo, total estimated integration error for probability flow

An error of 5.8 % is calculated on the probability of conflict using probability flow – a large overestimate on a probability of only 2.508 %. The true probability of conflict lies on the interval $[2.079, 2.937]$ % while the upper bound to the probability of conflict lies on the interval $[8.2238, 8.3922]$ %. The possible overestimate on the probability of conflict calculated is therefore bounded by the interval $[5.2868, 6.3132]$ %.

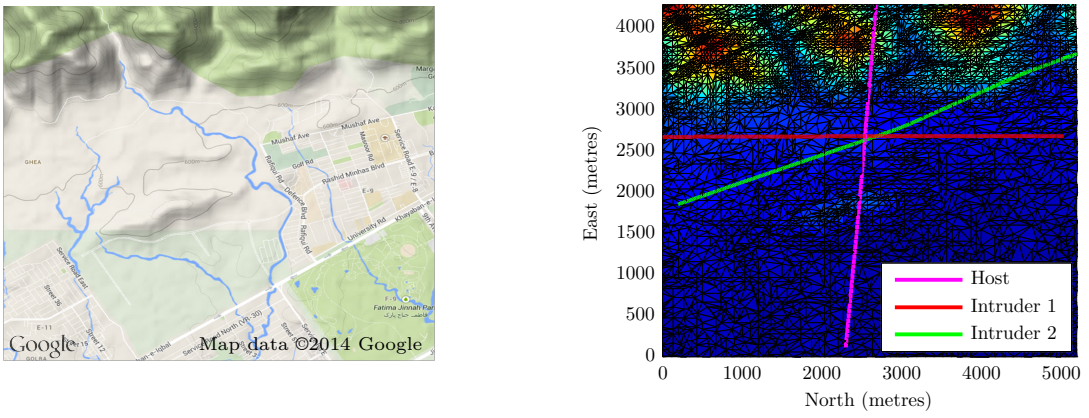
Because the overestimate is relative to the magnitude of risk, large probabilities of conflict will be accompanied by extremely large upper bounds. A large overestimate in this case would, however, not influence a resolution decision as evasive action would be required regardless. For small detected probabilities of conflict, a relatively large overestimate would make the difference between risk being below the alert threshold or above it. For example, in the results of table 5.5, a risk of 2.508 % would fall below a threshold of 5 % not warranting an alert, whereas the risk of 8.308 % calculated using probability flow would unnecessarily trigger a traffic advisory to the pilot.

From the information gathered in this section, it can be concluded that flying closely parallel to obstacle conflict regions leads to the over-accumulation of probability flow. This type of scenario is unlikely, but potentially very hazardous. It could, however, lead to false alarms near airports that operate parallel runways. This is also a common problem with TCAS for runway separations closer than 3000 ft [55]. As a solution, it may be possible

to use a linear combination of smaller Gaussian functions to represent the aircraft state PDF as a basis function. Propagation and accumulation of risk for a distorted state PDF can therefore be performed by instead propagating the individual Gaussian functions that make up the distorted PDF and accumulating risk from only those Gaussian functions that experience conflict. This solution is discussed in further detail in section 6.3 in chapter 5.

5.3 Scenario 3: Multiple Intruders and Terrain

Following the independent testing of the the probability flow algorithm in scenario 1 and 2 for a simple mid-air collision and CFIT collision respectively, the algorithm is applied to an example containing both of these aspects. The host aircraft in this scenario flies closely over the Margalla Hills near the Benazir Bhutto international airport, in Islamabad, Pakistan. This scenario was chosen in consideration of the Airblue flight 202 CFIT accident that occurred in the Margalla Hills in 2010. Additionally, two intruder aircraft closely crossing paths with the host aircraft are added to the simulation. If TCAS and EGPWS are to be replaced with a single system, potential new algorithms such as probability flow, should be evaluated on their ability to predict conflict with both terrain and multiple aircraft. The scenario trajectories are depicted in figure 5.15 along with a Google map of the area.



(a) Google Map (b) Top-View of Meshed Elevation Data

Figure 5.15: Terrain Maps of Margalla Hills, Islamabad

The host aircraft starts flying at 2.68 kilometres North and 4.2885 kilometres East at an altitude of 930 metres above sea level. The aircraft passes closely by a steep peak in the Margalla Hills at 12.5 seconds into the simulation before descending at an average velocity of 143 knots towards the airport. The first intruder aircraft crosses paths perpendicularly above the host aircraft 19.5 seconds into the simulation at a velocity of 140 knots. The second intruder aircraft crosses paths just below the host aircraft 20.25 seconds into the simulation at a velocity of 140 knots. A three-dimensional view of the scenario can be

seen in figure 5.16.

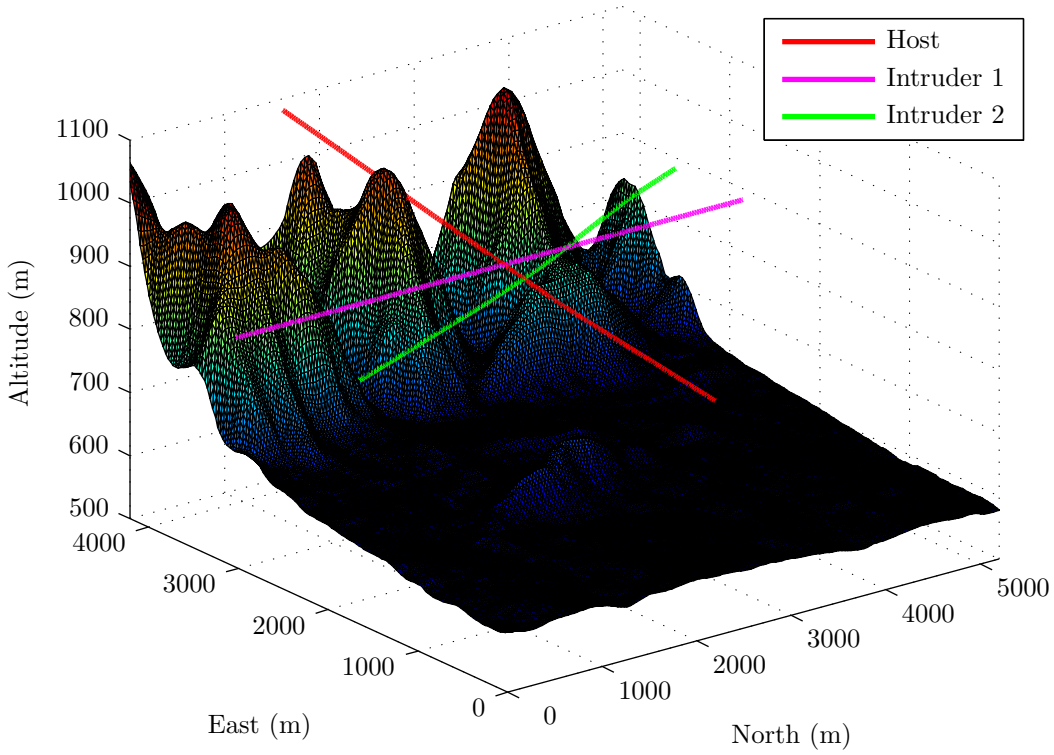
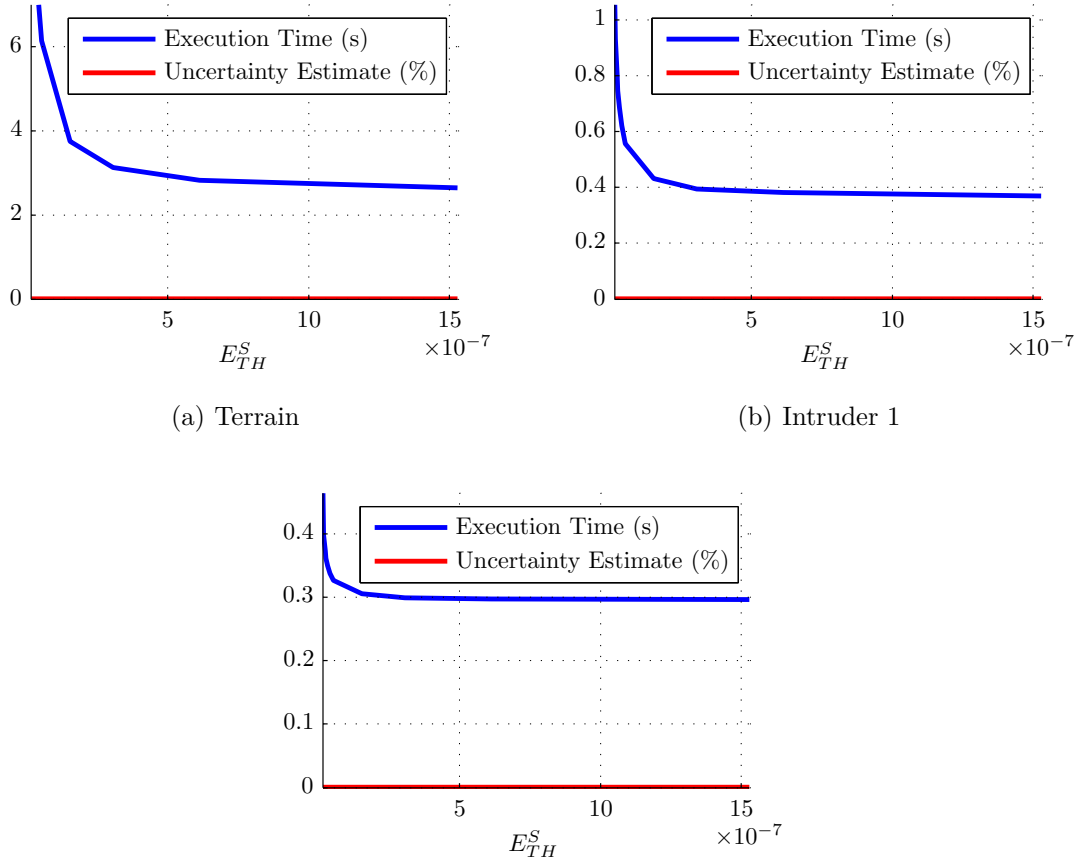


Figure 5.16: Scenario 3: 3D-View of Terrain and Aircraft Trajectories

Before applying any mesh reduction techniques, the effect of varying the surface integration error threshold E_{TH}^{S} over the interval $[2.3191 \times 10^{-8}, 2.3191 \times 10^{-5}]$ is measured (shown in figure 5.17). It can be clearly seen that changing the threshold has little effect on the uncertainty estimate for each calculated probability of conflict in this scenario, but the execution time of each calculation is significantly reduced for $5 \times 10^{-7} \leq E_{TH}^{S} \leq 2.3191 \times 10^{-5}$. Varying the time integration error threshold produces no significant effect on the uncertainty estimate or execution time and is therefore not shown here. This is once again due to the small sampling period used in simulation, ensuring negligible error between time samples.

As in scenario 2, when the probability flow method is implemented, only those mesh faces that lie within 3 standard deviations of the mean Monte Carlo trajectory are checked for conflict. Mesh reduction is also employed with insight from figures 5.18 and 5.19.

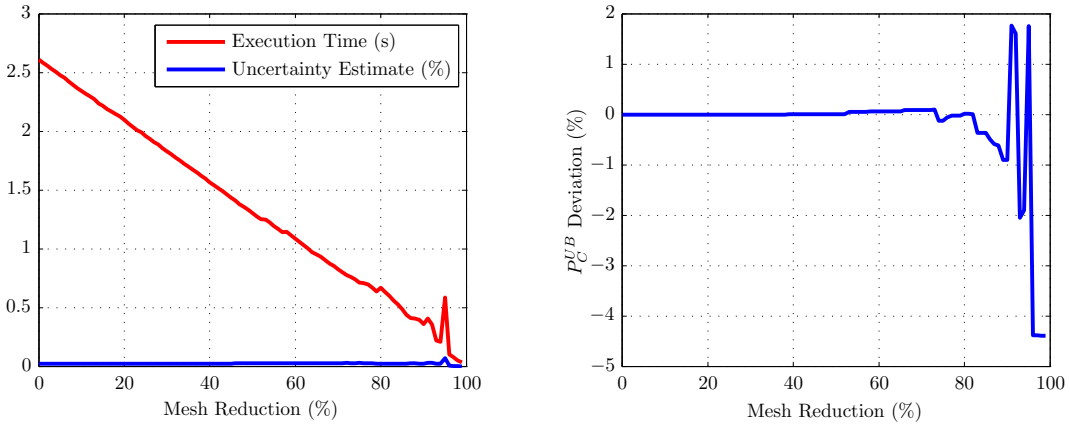


(a) Terrain

(b) Intruder 1

(c) Intruder 2

Figure 5.17: Scenario 3: Effect of Surface Integration Error Threshold on Execution Time and Uncertainty



(a)

(b)

Figure 5.18: Scenario 3: Effect of Mesh Reduction on Execution Time, Uncertainty and P_C^{UB}

Figure 5.18(a) shows the expected decline in execution time for large mesh reduction factors with a slight increase in total uncertainty. The upper bound to the probability of conflict also shows significant deviation for a mesh reduction between 79 and 100 %. Mesh

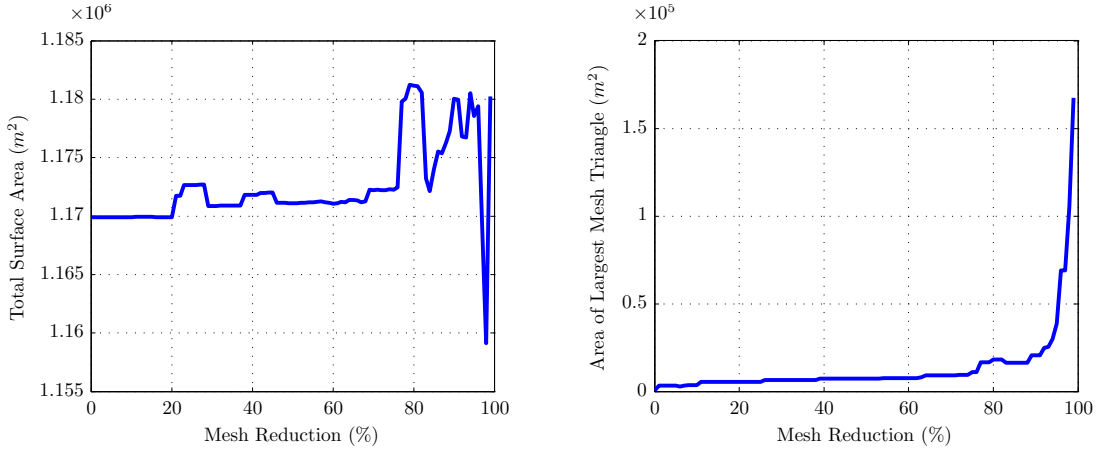


Figure 5.19: Scenario 3: Effect of Mesh Reduction on Surface Area and Maximum Triangle Size

reduction is therefore kept to 78 % resulting in an execution time of 0.68 seconds and a maximum triangle size of 15,561 m^2.

Table 5.6: Multiple Intruders and Terrain Simulation Results

	P_C (%)	Error* (%)	Uncertainty† (%)	Running Time (s)
Terrain				
Monte Carlo	2.8	-	0.4518	57988.19
Probability Flow	4.36	1.56	0.0242	0.68
Intruder 1				
Monte Carlo	0.63	-	0.2167	93432.37
Probability Flow	0.76	0.13	0.0193	0.4
Intruder 2				
Monte Carlo	0.19	-	0.1193	93811.2
Probability Flow	0.162	0.028	0.0059	0.32
Net				
Monte Carlo	3.593	-	0.51	245235.6
Probability Flow	5.241	1.648	0.0495	1.38

* compared to Monte Carlo simulation
† 3 × standard deviation for Monte Carlo, total estimated integration error for probability flow

The results of this scenario are indicated in table 5.6 on the previous page for each conflict event individually as well as for the associated net risk calculated using algorithm 3 on page 68. This scenario is interesting because each probability of conflict calculated with the probability flow method would not individually trigger a traffic advisory to the pilot for an alert threshold of 5 %, but collectively would indeed trigger an alert. Furthermore, the net "ground-truth" risk calculated using the Monte Carlo method would not trigger a TA, when the net result calculated using probability flow would.

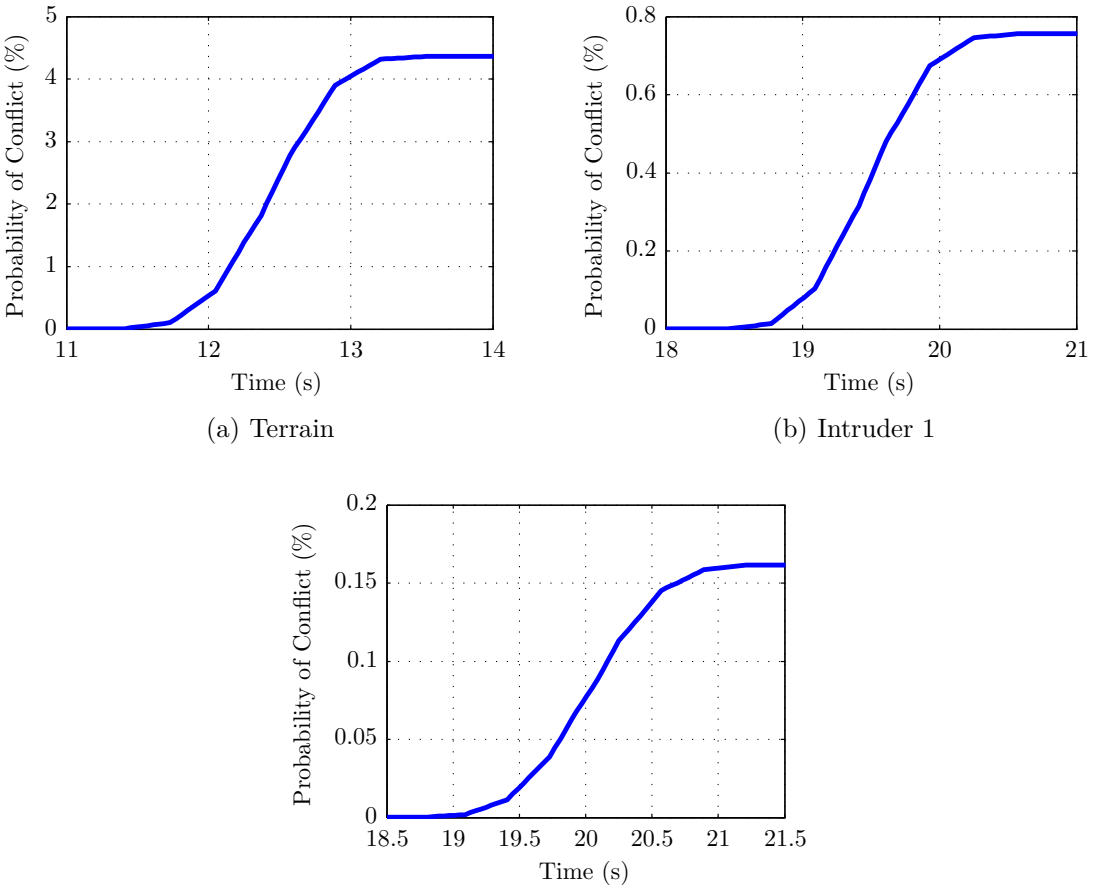
For the case of conflict with terrain, a standard deviation of 0.1506 % means that the true probability of conflict lies on the interval $[2.3482, 3.2518]$ %. For the conflict with intruder 1, a standard deviation of 0.0722 % means that the true probability of conflict lies on the interval $0.4133, 0.8467]$ %. Lastly, for the conflict with intruder 2, a standard deviation of 0.0398 % means that the true probability of conflict lies on the interval $[0.0707, 0.3093]$ %. The true net probability of conflict is therefore contained on the interval $[3.083, 4.103]$ %.

In comparison, the probability of conflict calculated using probability flow exhibits an error estimate of 0.0242 %, 0.0193 % and 0.0059 % for conflict with the terrain, first intruder and second intruder respectively. The corresponding upper bounded probabilities of conflict therefore lie on the intervals $[4.3358, 4.3842]$ %, $[0.7407, 0.7793]$ % and $[0.1561, 0.1679]$ %. The overestimates for these conflict events can then be said to lie on the intervals $[1.084, 2.036]$ %, $[0, 0.366]$ % and $[0, 0.0972]$ % respectively. The net risk calculated using probability flow is contained on the interval $[5.1915, 5.2905]$ % with an associated error on the interval $[1.0885, 2.2075]$ %.

The accumulation of the probability of conflict over time can be seen in figure 5.20 for each obstacle. From these graphs, it is seen that the predicted duration of conflict with the terrain is 1.8 seconds, with the first intruder is 1.9 seconds and with the second intruder is 2.25 seconds. This scenario, although unlikely, demonstrates that the probability flow method is capable of predicting conflict in a complex environment in a relatively short time (1.38 seconds).

5.4 Scenario 4: Aborted Landing

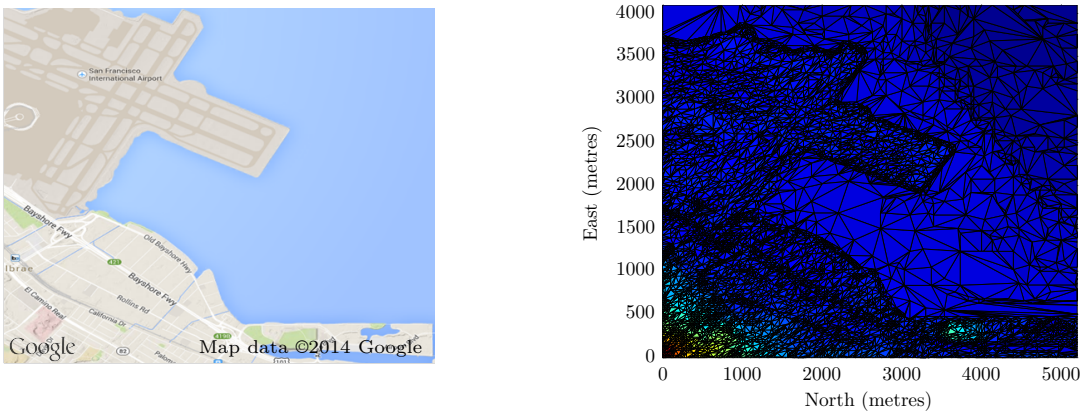
This scenario is modelled after the failed aborted landing that led to the crash of Asiana Airlines flight 214 on the runway at San Francisco international airport (SFO) in 2013. This scenario was chosen to test the performance of the probability flow algorithm in predicting a fairly common form of CFIT accident. A Google map and corresponding top-view of the meshed elevation data for the airport is shown in figure 5.21.



(a) Terrain (b) Intruder 1

(c) Intruder 2

Figure 5.20: Scenario 3: Accumulation of Risk over Time for Each Obstacle



(a) Google Map (b) Top-View of Meshed Elevation Data

Figure 5.21: Terrain Maps of San Francisco International Airport

The host aircraft descends at an average velocity of 148 knots towards the runway. At 19.38 seconds into the simulation, the aircraft aborts the landing at the beginning of the runway and begins to ascend again for a go-around manoeuvre. The scenario is shown

more in clearly in three-dimensions in figure 5.22.

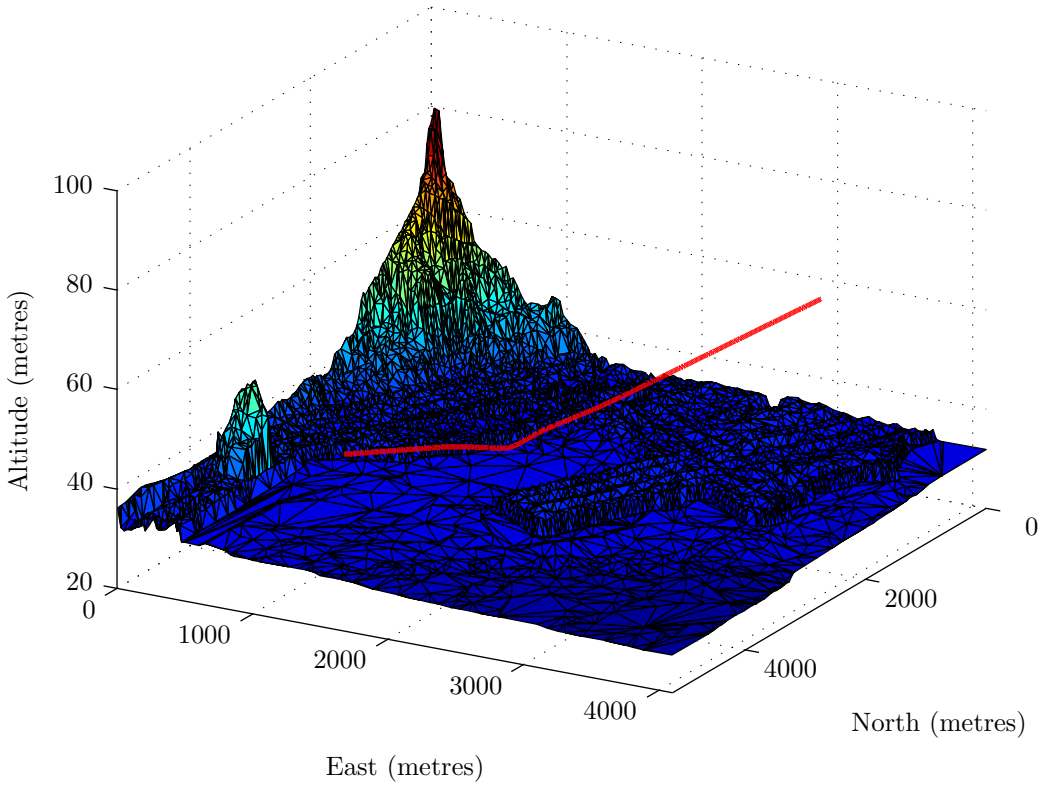


Figure 5.22: Scenario 4: 3D of Terrain and Aircraft Trajectory

In airport landing scenarios, it is difficult to decide how the runway should be handled in the conflict detection module. The intent of the aircraft is to make contact with the runway, which would result in the accumulation of an associated probability of conflict. To resolve this problem, it is possible to remove the terrain mesh faces that lie exactly on the runway. However, in an aborted landing scenario, any contact with the runway must be seen as a conflict. The mesh faces that make up the runway are therefore kept in the elevation map for this scenario. Mesh reduction techniques are, however, applied as they are in scenarios 1, 2 and 3.

The effect of varying the surface integration error threshold over the interval $[1.2864 \times 10^{-7}, 1.2864 \times 10^{-4}]$ for this scenario is shown in figure 5.23. It is observed that the execution time tends towards 0.41 seconds for $E_{TH}^{S} \geq 4 \times 10^{-6}$ and no effect can be seen on the uncertainty estimate. The surface integration error threshold is therefore constant kept at $E_{TH}^{S} = 1.2864 \times 10^{-7}$ for this scenario.

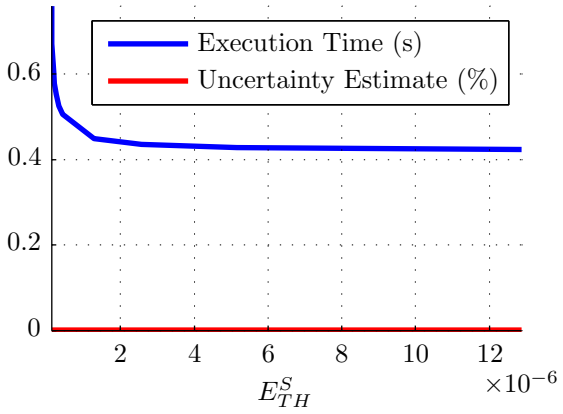


Figure 5.23: Scenario 4: Effect of Surface Integration Error Threshold on Execution Time and Uncertainty

Because the aircraft aborts the landing at 21 metres above the runway, excluding the terrain mesh faces that lie outside 3 standard deviations of the mean Monte Carlo trajectory means that only a small strip of the runway needs to be checked for conflict. As a result, reducing the number of mesh faces using MATLAB's `reducepatch` function needs to be used sparingly to prevent the remaining mesh faces from becoming too large, distorting the original shape of the terrain. The effect of varying the mesh reduction factor is shown in figure 5.24 and 5.25.

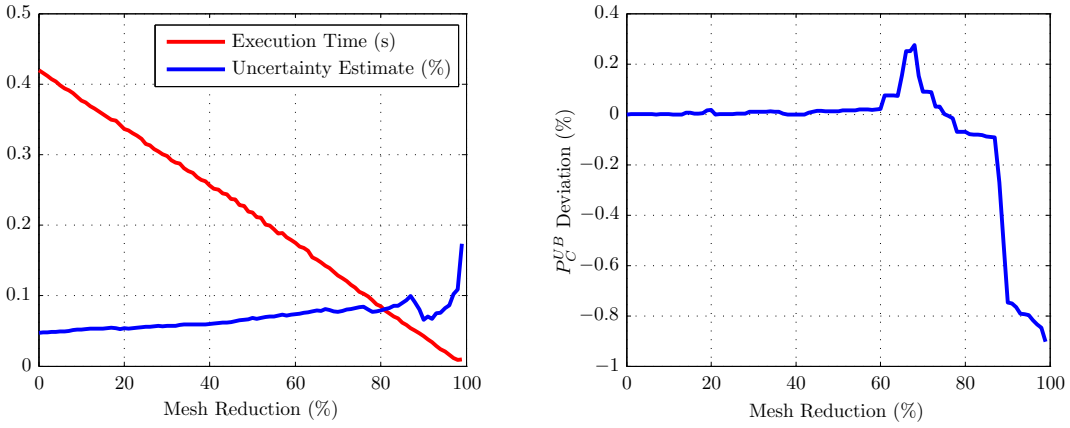


Figure 5.24: Scenario 4: Effect of Mesh Reduction on Execution Time, Uncertainty and P_C^{UB}

A linear decline in computation time is seen in figure 5.24(a) for increasing percentages of mesh reduction with a slight increase in the estimate of uncertainty on P_C^{UB}. In figure 5.24(b), a large deviation in the calculated probability of conflict is observed for mesh reduction above 60 %. Furthermore, the total mesh surface area suffers a sudden decline

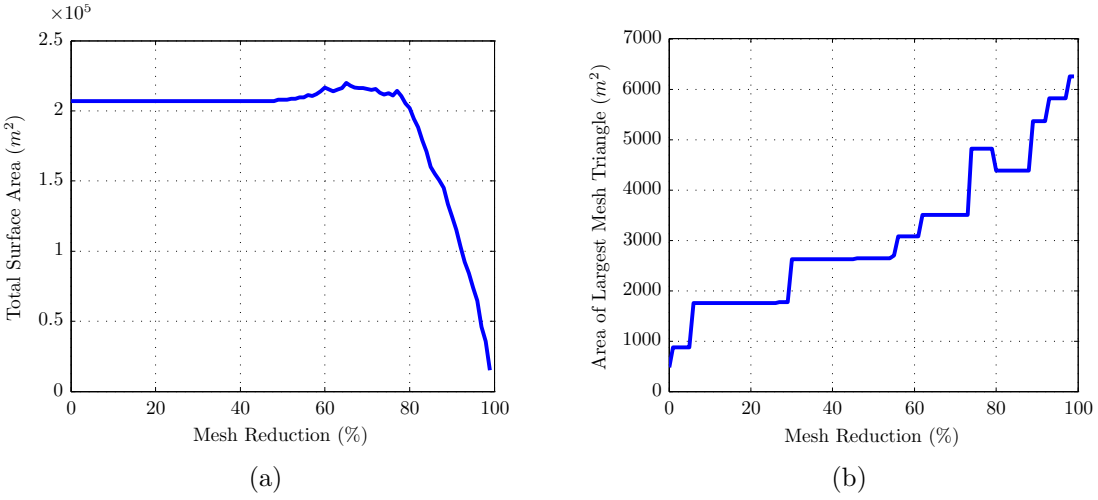


Figure 5.25: Scenario 4: Effect of Mesh Reduction on Surface Area and Maximum Triangle Size

for reductions larger than 60 % as seen in figure 5.25(a), reflecting a distortion in the original shape of the terrain. This scenario therefore warrants to use of a maximum mesh reduction of 60 %, ensuring that the estimate of uncertainty remains below 0.1 % while still achieving a reduced computation time of 0.19 seconds. The maximum allowable triangle size in the mesh pertaining to this scenario is 3079.4 m^2.

The results of the aborted landing scenario are listed in table 5.7. Due to a standard deviation of 0.071 % on the Monte Carlo probability of conflict, the true probability of conflict lies on the interval $[0.404, 0.83]$ %. The uncertainty of 0.0428 % means that the upper bound to the probability of conflict lies on the interval $[1.8142, 1.8998]$ %. The error estimate on the true probability of conflict obtained using probability flow is therefore on the interval $[0.9842, 1.4958]$ %.

Table 5.7: Aborted Landing Simulation Results

Method	P_C (%)	**Error*** (%)	**Uncertainty†** (%)	**Running Time (s)**
Monte Carlo	0.617	-	0.213	51233.31
Probability Flow	1.857	1.24	0.0428	0.19

* compared to Monte Carlo simulation
† 3 × standard deviation for Monte Carlo, total estimated integration error for probability flow

When the error estimate is larger than the probability of conflict itself, the overestimate can especially be considered unreasonably large. As in scenario 2, the accumulation of probability flow briefly occurs parallel to the conflict region i.e. the runway, and a large

overestimate results. In this case, the overestimate is not large enough to change whether or not a warning is issued to the pilot, but the importance of fixing this problem in the probability flow algorithm is evident, particularly for small predicted probabilities.

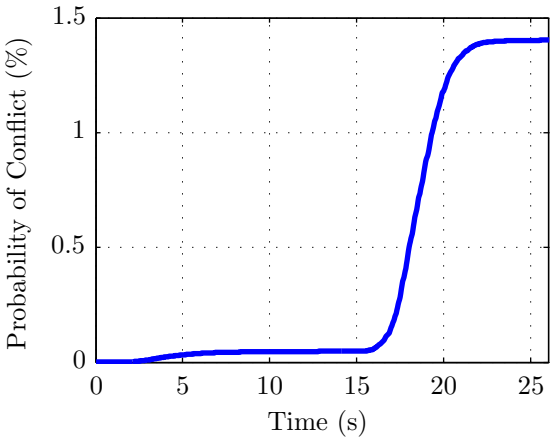


Figure 5.26: Scenario 4: Accumulation of Risk over Time

Figure 5.26 shows the accumulation of conflict over time using probability flow. The trend is different to that of previous scenarios in that a very small conflict is detected 4 seconds into the simulation before the majority of probability is accumulated between 16 and 22 seconds. The initial conflict could correspond to the slight risk of colliding with the sea wall that separates the runway from the Pacific Ocean. The second conflict corresponds to the risk of colliding with the runway at the instant of the aircraft's lowest altitude when the command to climb is issued.

Chapter 6

Conclusion

This chapter contains a summary of the results obtained in this thesis, a list of the most important observations regarding the probability flow method and a few recommendations for improvement of the algorithm.

6.1 Summary

This thesis describes the evaluation of the probability flow algorithm as a method of conflict detection in uncertain, cluttered, airport environments containing multiple dynamic obstacles and hazardous terrain. In chapter 1, the general problem description, expected results, significance and objectives of this thesis are outlined. This section summarises how these objectives are met in each chapter of the thesis.

Through the review of literature in chapter 2, an understanding is achieved of the airport environment, air traffic management systems and existing collision avoidance systems. Additionally, various key methods that led to the development of the probability flow algorithm are discussed in order to provide context and motivation as to why this algorithm forms the primary focus of this thesis. To the author's knowledge, it was found that only the Monte Carlo and probability flow methods are capable of handling such a general problem description of the aviation environment, containing multiple non-linear aircraft systems, terrain, uncertainty and intent.

Chapter 3 describes the modelling of all relevant aspects of the airport environment. First, a framework describing a collision avoidance system incorporating probability flow is proposed. The conflict region of each aircraft is chosen to be spherical so as to avoid repeated computation of the Minkowski sum as a result of uncertain changes in orientation at each time step in the future. The idea of generating terrain data using the Google elevation API is introduced and the motivation behind the selection of a Delaunay-based method of triangulation is also discussed. Chapter 3 is concluded with the definition of the principal and inertial axis systems used throughout the thesis as well as an introduction to the

generic 6 degrees-of-freedom aircraft model.

The implementation of the modelling concepts discussed in chapter 3 as well as the implementation of probability flow and Monte Carlo methods is detailed in chapter 4. A brief overview of the software structure is provided along with a discussion on the usage limits and format of the HTTP requests used to generate terrain maps with the Google elevation API. The chapter goes on to describe how disturbance is introduced into a Simulink aircraft model to imitate uncertainty present in trajectory propagation before detailing the implementation of cross-track and altitude controllers in the model.

Furthermore, chapter 4 discusses the initialisation of the aircraft model for effective Monte Carlo simulation and the mathematical definition of the probability flow method is given. The midpoint rule and Simpson's rule are used to adaptively integrate the surface and time integrals respectively present in the definition of probability flow. Problems were encountered with insufficient dynamic memory in the generation of thousands of Monte Carlo trajectories, but the problem is resolved by writing data to .bin files on disk. Lastly, this chapter presents the novel application of the inclusion-exclusion principle for the calculation of the net risk in scenarios containing multiple obstacles. It was found that the net probability of conflict calculated using the probability flow method will still contain an upper bound if at least one obstacle conflict probability in the scenario also contains an upper bound.

The results of performing conflict detection with probability flow and the Monte Carlo method in 4 different simulated flight scenarios is presented in chapter 5. Each scenario is chosen to highlight different strengths and weaknesses of Van Daalen's probability flow method. The two-aeroplane example demonstrates that the method is capable of performing accurate prediction of conflict in less than 1 second with little or no uncertainty for scenarios in which the host and intruder aircraft diverge quickly after conflict is initially detected. The probability flow method therefore has the potential to detect mid-air collisions with the same efficiency as TCAS, but with improved accuracy due to the incorporation of intent and uncertainty.

The terrain-only scenario in chapter 5 illuminates a shortcoming of the probability flow method in that flying parallel to an obstacle conflict region results in a large overestimate on the probability of conflict relative to the Monte Carlo simulation result. Moreover, the longer the duration of conflict, the larger the overestimate will be. Aside from this shortcoming, the algorithm still executes quickly for large areas of terrain with the assistance of mesh reduction techniques. Mesh faces that lie outside of 3 standard deviations of

the mean Monte Carlo trajectory are disregarded and the remaining mesh faces are effectively simplified using MATLAB's `reducepatch` function, thus reducing computation time.

Scenario 3 in chapter 5 demonstrates the ability of the probability flow method to accurately handle scenarios containing multiple intruder aircraft as well as terrain. Although conflict detection could not be performed in under 1 second as with the previous scenarios, an execution time of 1.38 seconds for such a complex scenario with a look-ahead time of 42 seconds is reasonable. This scenario also successfully implements the recursive inclusion-exclusion principle algorithm as discussed in chapter 4 to calculate net risk along the host aircraft's path. The importance of this function is highlighted in that individually each conflict event would not trigger an alert, but collectively would have resulted in a traffic advisory to the pilot. Scenario 3 therefore clearly demonstrates that the probability flow method could potentially perform the same functions as TCAS and EGPWS, but as a single, integrated system. In this way, risk can be prioritised based on urgency rather than the type of obstacle the risk is associated with. Furthermore, this scenario makes it clear that large overestimates on the probability of conflict are not obtained in mid-air collision scenarios, but only in scenarios involving large expanses of terrain where the aircraft spends a relatively long time in the vicinity of the conflict region.

The final scenario in chapter 5 deals with an aborted landing at an airport. Like scenario 2, a large overestimate is obtained on a small probability of conflict due to the brief accumulation of probability flow closely parallel to the airport runway. This scenario further stresses the need for a solution to this problem if probability flow is to be successfully implemented in an airport environment. Even though a large overestimate is present on the probability of conflict, this scenario still demonstrates that probability flow is capable of calculating risk associated with an aborted landing situation fairly quickly.

Analysis of MATLAB's `reducepatch` function in scenarios 2, 3 and 4 shows that the terrain data can be simplified to improve execution time, but the degree of possible mesh reduction without compromising accuracy is dependent on the trend of the terrain and type of scenario. Determining the optimal mesh reduction factor specific to each scenario is an avenue of further research that would prove beneficial.

In conclusion, the probability flow method is a viable option for real-time probabilistic conflict detection in the aviation context. Probability flow can accurately and quickly predict brief mid-air and CFIT conflicts for a look-ahead time of up to 60 seconds. Using probability flow allows flexibility in choosing the size of the conflict region and the range at which conflict is detected.

6.2 Probability Flow Observations

- Conflicts between aircraft are predicted more quickly than conflicts with the terrain, but CFIT accidents are more prevalent than mid-air collisions.
- The number of triangles in the terrain mesh have a large influence on the algorithm execution time and accuracy of the estimated risk.
- For a very small simulation sampling period, the size of the time integration error threshold plays little role in the result uncertainty and execution time of the algorithm.
- A very small sampling period (≤ 0.01) is necessary in the aviation context to ensure that conflict is not missed between samples for aircraft that travel at high speeds.
- Large overestimates on the probability of conflict are only obtained in scenarios in which the aircraft spends a large amount of time in the vicinity of the conflict region. This is particularly prevalent in CFIT scenarios containing large expanses of terrain rather than mid-air collision scenarios where the aircraft conflict region is small.

6.3 Future Work

In this section, recommendations for the improvement of the probability flow method are proposed. Considering the practical observations discussed in the previous section, further work is required before this algorithm could potentially be implemented as part of a legitimate air traffic management system.

1. **Flying Parallel to Obstacles**

   The large overestimate that results from flying parallel to an obstacle surface boundary needs to be reduced to ensure a tight upper bound on the probability of conflict at all times. Flying parallel to a mountain or building is very dangerous and should trigger a warning or resolution advisory regardless, but this problem needs to be resolved if the occurrence of false alarms at airports operating parallel runways is to be reduced.

   Alternatively, a first order second moment approximation can be obtained of the distorted aircraft state PDF after the accumulation of probability flow has taken place. It can be assumed that the distorted PDF is still a Gaussian function with a new mean and variance. Given that the original aircraft state PDF is described by

the random variable $X \sim (\overline{x}, \sigma_X^2)$ and $Y = \mathrm{g}(X)$ is a generic non-linear function of X representing the distorted aircraft state PDF, the mean $\overline{y}$ and variance σ_Y^2 can be found. First, a linear Taylor expansion around $\overline{x}$ is used to replace $\mathrm{g}(X)$ with a linear function of X [56].

$$Y = \mathrm{g}(X) \approx \mathrm{g}(\overline{x}) + \left.\frac{\mathrm{dg}(X)}{\mathrm{d}X}\right|_{\overline{x}} (X - \overline{x}) \tag{6.3.1}$$

The approximate mean value of the distorted PDF is then $\overline{y} = \mathrm{g}(\overline{x})$ and the approximate variance of the distorted PDF is given by

$$\sigma_Y^2 = \left(\left.\frac{\mathrm{dg}(X)}{\mathrm{d}X}\right|_{\overline{x}} \right)^2 \sigma_X^2. \tag{6.3.2}$$

The problem with this method is that knowledge of the aircraft's dynamics is required to formulate $\mathrm{g}(\,\cdot\,)$. With the probability flow method, it assumed that pre-existing data describing the mean and covariance along the predicted trajectory is readily available, thus eliminating the need for knowledge of the aircraft's dynamics. Furthermore, calculating $\mathrm{g}(\,\cdot\,)$ at each time step of propagation parallel to an obstacle will slow down execution time considerably.

The best solution is possibly to describe the aircraft state PDF as a sum of smaller Gaussian functions. Each Gaussian function can then be propagated independently and summed before being checked for conflict at each time step. The integral of a Gaussian function curve is $H_X|\sigma_{X_n}|\sqrt{\pi}$ where H_X is the height of the curve and σ_{X_n} is the standard deviation. Before conflict, the integral of the sum of all the Gaussian functions must be equal to 1 – the value of the integral of the normal distribution describing the aircraft state PDF. After a conflict has been experienced, however, the probability that has flowed through the conflict region surface boundary is accumulated and removed from the PDF. The resulting probability of conflict is essentially the integral of the sum the individual Gaussian functions that experienced conflict.

The remaining Gaussian functions can then easily be individually propagated further, while collectively still accurately representing the distorted aircraft state PDF. The number of Gaussian functions used to approximate the aircraft state PDF will no doubt influence the execution time and should be kept as low as possible. In practice, the Gaussian functions can be represented using a basis function expansion of a Gaussian function describing the aircraft state PDF.

2. **Gaussian Assumption**

The most significant assumption made in the probability flow method, is that the position and velocity aircraft state PDFs are jointly Gaussian. Scenarios may, however, arise in which the true position and velocity distributions are not Gaussian. Future work could include a study of instances in which this would be the case and the effect this has on the ability of the probability flow method to predict conflict with a relatively tight margin of error.

In this thesis, it was found that operating the aircraft model beyond its aerodynamic constraints, something which may be required in an emergency resolution scenario, resulted in the deviation of the aircraft position PDF from its Gaussian form. The resulting probability of conflict was therefore observed to be underestimated – a crucial flaw if present in an aircraft conflict detection system. Moreover, it may be possible to derive a formulation of probability flow for other types of distributions using the sum of Gaussians method described previously in point 1.

3. **Mean and Covariance**

This thesis assumes knowledge of the position and velocity mean and covariance along the future trajectory and uses Monte Carlo simulation to calculate these values. Additional research into the field of efficient and accurate trajectory propagation could assist in the characterisation of future uncertainty for varying aerodynamic modes, manoeuvres and environmental factors.

Ideally, accurate trajectory propagation incorporating uncertainty and intent should be performed in real-time. Alternatively, this research could assist in the creation of a generic look-up table specific to particular aircraft models. A comparison between results obtained using the look-up table and using Monte Carlo simulation could provide a measure of the viability of using pre-defined values of uncertainty for different situations.

4. **Integration Rules**

A large bottleneck exists in the execution of the surface integral in equation 4.5.1 on page 61. Currently, the midpoint rule is used to adaptively estimate the surface integral, but it is possible that other integration rules in the Newton-Cotes family may be more efficient. Future work could include the testing of the trapezoidal rule,

Milne's rule and Boole's rule to name a few.

5. **Mesh Reduction**

   In this thesis, the MATLAB `reducepatch` function was used to reduce the number of triangles used to represent the terrain mesh, thus reducing computation time. The use of more sophisticated mesh reduction techniques should be investigated to further improve efficiency without compromising accuracy in prediction.

Bibliography

[1] National Aeronautics and Space Administration: Air Traffic Management System. http://virtualskies.arc.nasa.gov/atm/2.html, Apr 2013.

[2] Kuchar, J.K. and Yang, L.C.: Using Intent Information in Probabilistic Conflict Analysis. *American Institute of Aeronautics and Astronautics, Inc.*, vol. A98, no. 37082, pp. 797 – 806, 1998.

[3] Wayne Gallo, D.T.: Traffic Alert and Collision Avoidance System (TCAS). FAA Flight Standards Pilot Outreach Program Presentation, Jun 2012.

[4] Jones, T.: *Real-time Probabilistic Collision Avoidance for Autonomous Vehicles, using Order Reductive Conflict Metrics*. Ph.D. thesis, 2003.

[5] Peddle, I.K.: Introductory Course to Aircraft Dynamics. Advanced Automation 833 Course Notes, April 2013.

[6] Federal Aviation Administration: Re-categorization (RECAT) of FAA Wake Turbulence Separation Categories at Specific Airports. Tech. Rep., U. S. Department of Transportation, 2013.

[7] Federal Aviation Administration: *Pilot's Handbook of Aeronautical Knowledge*. Faa-h-8083-25a edn. Flight Standards Service, 2008.

[8] Federal Aviation Administration (FAA): Introduction to TCAS II, Version 7.1. Tech. Rep., U.S. Department of Transport, 2011.

[9] European Aviation Safety Agency: Mode S Transponder in High Density Operational Environment. Tech. Rep., European Aviation Safety Agency, 2010.

[10] Kuchar, J.K. and Yang, L.C.: A Review of Conflict Detection and Resolution Modelling Methods. *IEEE Transactions on Intelligent Transportation Systems*, vol. 1, no. 4, pp. 179 – 189, 2000.

[11] International Civil Aviation Organization: Procedures for Air Traffic Management. Tech. Rep., International Civil Aviation Organization, 2007.

[12] Trani, A.A.: Aircraft Classifications. Air Transportation Systems Course Notes, July 2014.

[13] Van Daalen, C.E.: *Conflict Detection and Resolution for Autonomous Vehicles*. Ph.D. thesis, 2010.

[14] Paielli, R. and Erzberger, H.: Conflict Probability Estimation for Free Flight. *AIAA Journal of Guidance, Control and Dynamics*, vol. 20, no. 3, pp. 588 – 596, 1997.

[15] Marco Poretta, Wolfgang Schuster, A.M.W.O.: Strategic Conflict Detection and Resolution Using Aircraft Intent Information. *The Journal of Navigation*, vol. 63, no. 1, pp. 61–88, 2010.

[16] International Civil Aviation Organization: ICAO Maps and Web Applications for the Aeronautical World. http://gis.icao.int/gallery/index.html, Apr 2013.

[17] European Organisation for the Safety of Air Navigation: Principles of Mode S Operation and Interrogator Codes. Tech. Rep., Eurocontrol, 2003.

[18] International Civil Aviation Organization: Manual on Mode S Specific Services. Tech. Rep., International Civil Aviation Organization, 2004.

[19] International Civil Aviation Organization: Airborne Collision Avoidance System (ACAS) Manual. Tech. Rep., International Civil Aviation Organization, 2006.

[20] Ruley, J.D.: Zaon PCAS XRX (Portable Collision Avoidance with Direction). Plane and Pilot Magazine, Feb 2008.

[21] OCAS Incorporated: What is OCAS? http://www.ocasinc.com/, Sep 2014.

[22] SKYbrary: Automatic Dependent Surveillance Broadcast (ADS-B). http://www.skybrary.aero/index.php, May 2013.

[23] ADS-B Technologies: What is ADS-B? http://www.ads-b.com/default.htm, May 2013.

[24] Trig Avionics Limited: Ads-b and next-gen avionics. http://www.trig-avionics.com/adsb.html, May 2013.

[25] Wangermann, J.P. and Stengel, R.F.: Principled Negotiation between Intelligent Agents: a Model for Air Traffic Management. *Artificial Intelligence in Engineering*, vol. 12, pp. 177 – 187, 1998.

[26] Jones, T.: Tractable Conflict Risk Accumulation in Quadratic Space for Autonomous Vehicles. *AIAA Journal of Guidance, Control and Dynamics*, vol. 29, no. 1, pp. 39 – 48, 2006.

[27] Bureau of Aircraft Accidents Archives (B3A): Statistics: Crash Rate per Year. http://www.baaa-acro.com/general-statistics/crashs-rate-per-year/, Jul 2014.

[28] Boeing: Boeing training aid addresses leading accident cause. Press Release, Feb 1997.

[29] Maria Prandini, O.W.: Distributed Control and Stochastic Analysis of Hybrid Systems Supporting Safety Critical Real-Time Systems Design. *IST-2001-32460*, vol. D3.2, no. 1.2, 2005.

[30] Sarangi, S.: *Surface Reconstruction from Unorganized Point Cloud Data Using Incremental Delaunay Triangulation*. Isbn: 0549369279, 9780549369271 edn. ProQuest, 2007.

[31] Edelsbrunner, H. and Mücke, E.P.: Three-Dimensional Alpha Shapes. *ACM Transactions on Graphics*, vol. 13, no. 1, pp. 43–72, 1994.

[32] Amenta, N., Bern, M. and Kamvysselis, M.: A New Voronoi-Based Surface Reconstruction Algorithm. In: *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pp. 415–421. ACM, 1998.

[33] Dey, T.K. and Ray, T.: Polygonal Surface Remeshing with Delaunay Refinement. *Engineering with Computers*, vol. 26, no. 3, pp. 289–301, 2010.

[34] Kós, G.: An Algorithm to Triangulate Surfaces in 3D using Unorganised Point Clouds. In: *Geometric Modelling*, pp. 219–232. Springer Vienna, 2001.

[35] Kuo, C.-C. and Yau, H.-T.: A Delaunay-Based Region-Growing Approach to Surface Reconstruction from Unorganized Points. *Computer-Aided Design*, vol. 37, no. 8, pp. 825–835, 2005.

[36] Jean-Daniel Boissonnat, Olivier Devillers, S.P.M.T.M.Y.: Triangulations in CGAL. *Computational Geometry: Theory and Applications*, vol. 22, no. SoCG00, pp. 5–19, 2002.

[37] Cheng, S., Dey, T. and Shewchuk, J.: *Delaunay Mesh Generation*. Chapman & Hall/CRC Computer & Information Science Series. Taylor & Francis, 2012. ISBN 9781584887300.

[38] Mencl, R. and Muller, H.: Interpolation and Approximation of Surfaces from Three-Dimensional Scattered Data Points. In: *Scientific Visualization Conference, 1997*, pp. 223–223. IEEE, 1997.

[39] Bernardini, F., Mittleman, J., Rushmeier, H., Silva, C. and Taubin, G.: The Ball-Pivoting Algorithm for Surface Reconstruction. *Visualization and Computer Graphics, IEEE Transactions on*, vol. 5, no. 4, pp. 349–359, 1999.

[40] Gopi, M. and Krishnan, S.: A Fast and Efficient Projection-Based Approach for Surface Reconstruction. In: *Computer Graphics and Image Processing, 2002. Proceedings. XV Brazilian Symposium on*, pp. 179–186. IEEE, 2002.

[41] Cohen-Steiner, D. and Da, F.: A Greedy Delaunay-Based Surface Reconstruction Algorithm. *The Visual Computer*, vol. 20, no. 1, pp. 4–16, 2004.

[42] Hoppe, H., DeRose, T., Duchamp, T., McDonald, J. and Stuetzle, W.: *Surface Reconstruction from Unorganized Points*, vol. 26. ACM, 1992.

[43] Curless, B. and Levoy, M.: A Volumetric Method for Building Complex Models from Range Images. In: *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pp. 303–312. ACM, 1996.

[44] Bajaj, C.L., Bernardini, F. and Xu, G.: Automatic Reconstruction of Surfaces and Scalar Fields from 3D Scans. In: *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pp. 109–118. ACM, 1995.

[45] Boissonnat, J.-D. and Cazals, F.: Smooth Surface Reconstruction via Natural Neighbour Interpolation of Distance Functions. In: *Proceedings of the sixteenth annual symposium on Computational geometry*, pp. 223–232. ACM, 2000.

[46] Bolle, R.M. and Vemuri, B.C.: On Three-Dimensional Surface Reconstruction Methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 1, pp. 1–13, 1991.

[47] Skiena, S.: *The Algorithm Design Manual*. Illustrated edn. Springer Science and Business Media, 2009.

[48] Molinari, M.: XML Toolbox. MATLAB File Exchange, Apr 2005.

[49] Mathworks: Dryden Wind Turbulence Model (Continuous). Online Help Documentation, Sep 2014.

[50] Mathworks: Von Kármán Wind Turbulence Model (Continuous). Online Help Documentation, Sep 2014.

[51] O'Rourke, J.: *Computational Geometry in C*. Cambridge University Press, oct 1998.

[52] Kahaner, D., Moler, C., Nash, S. and Forsythe, G.: *Numerical Methods and Software*. Prentice-Hall series in computational mathematics. Prentice Hall, 1989.

[53] Mazur, D.: *Combinatorics: A Guided Tour*. MAA textbooks. Mathematical Association of America, 2010. ISBN 9780883857625.

[54] Kuchar, J.E. and Drumm, A.C.: The Traffic Alert and Collision Avoidance System. *Lincoln Laboratory Journal*, vol. 16, no. 2, p. 277, 2007.

[55] Carpenter, B.D. and Kuchar, J.K.: Probability-Based Collision Alerting Logic for Closely-Spaced Parallel Approach. 1997.

[56] Veneziano, D.: Brief Notes 6, First-Order Second-Moment Propagation of Uncertainty. Course Notes, Probability and Statistics in Engineering 1-151, Massachusetts Institute of Technology, 2005.